BlackBerry Java Application UI and Navigation

Version: 5.0

Entwicklungshandbuch

BlackBerry

Veröffentlicht: 2010-06-07
SWD-810476-0607021351-003

Inhaltsverzeichnis

1 Erstellen einer Benutzeroberfläche, die mit normalen BlackBerry-Benutzeroberflächen kompatibel ist 5

2 BlackBerry-Gerät: Eingabe und Navigation für Benutzer 6
Touchscreen 6
Trackball oder Trackpad 8
Trackball-Empfindlichkeit 8
Trackball-Bewegung 9
Trackwheel 9
Tastatur 9
Interaktionsmethoden auf BlackBerry-Geräten 10

3 Bildschirme 11
Screen-Klassen 11
Verwalten eines Zeichenbereichs 12
Erstellen eines Bildschirmübergangs 13
Codebeispiel: Erstellen eines Bildschirmübergangs 15
Festlegen des Formats und der Richtung des Bildschirms 16
Abrufen der Ausrichtung des Touchscreen 17
Schränken Sie die Richtung des Touchscreens ein 17
Benachrichtigen lassen, wenn sich die Größe des Zeichenbereichs auf dem Touchscreen ändert 18

4 Beschleunigungssensor 19
Arten von Beschleunigungssensor-Daten 19
Abrufen von Beschleunigungssensordaten 19
Abrufen der Beschleunigungssensordaten in bestimmten Abständen 20
Abfragen des Beschleunigungssensors, wenn sich die Anwendung im Vordergrund befindet 21
Abfragen des Beschleunigungssensors, wenn sich die Anwendung im Hintergrund befindet 22
Speichern der ausgelesenen Beschleunigungssensorwerte in einem Puffer 22
Abrufen von Beschleunigungssensordaten aus einem Puffer 23
Ermitteln des Zeitpunkts des Abrufens eines Werts vom Beschleunigungssensor 23

5 Ereignisse 25
Reagierenauf Navigationsereignisse 25
Bestimmen der Art der Eingabemethode 25
Reagieren auf Touchscreen-Ereignisse 26

Antworten auf Systemereignisse, während der Benutzer den Bildschirm berührt ... 27
Reagieren auf einen Benutzer, der einen Finger schnell nach oben über den Bildschirm bewegt ... 27
Reagieren auf einen Benutzer, der einen Finger schnell nach unten über den Bildschirm bewegt ... 28
Reagieren auf einen Benutzer, der einen Finger schnell nach rechts über den Bildschirm bewegt ... 29
Reagieren auf einen Benutzer, der schnell einen Finger nach rechts über den Bildschirm bewegt ... 30
Reagieren auf einen Benutzer, der auf den Bildschirm klickt ... 30
Reagieren auf einen Benutzer, der den Bildschirm zweimal schnell berührt ... 31
Reagieren auf einen Benutzer, der ein Element auf dem Bildschirm berührt und zieht ... 32
Reagieren auf einen Benutzer, der den Bildschirm leicht berührt ... 32
Reagieren auf eine Bildlaufaktion ... 33
Reagieren auf einen Benutzer, der den Bildschirm an zwei Stellen gleichzeitig berührt ... 34
Ereignisinjektion ... 34

6 Anordnen von Benutzeroberflächenkomponenten ... 36
Anordnen von Benutzeroberflächenkomponenten ... 37
Erstellen eines Raster-Layouts ... 38
Erstellen einer Gittergröße ... 39
Anzeigen von Feldern in einem temporären Paar von Managern ... 42
Vorübergehendes Anzeigen eines ButtonField und eines LabelField oben bzw. unten am Bildschirm ... 42

7 Benutzeroberflächen-Komponenten ... 45
Hinzufügen einer Benutzeroberflächenkomponente zu einem Bildschirm ... 45
Ausrichten eines Feldes an einer Textzeile ... 45
Autocomplete-Textfeld ... 45
Erstellen eines autocomplete-Textfelds aus einem Datensatz ... 46
Erstellen eines autocomplete-Textfelds aus einer Datenquelle ... 48
Verwenden von Datenquellen und Feldern mit einem autocomplete-Textfeld ... 50
Schaltflächen ... 52
Bewährtes Verfahren: Implementieren von Schaltflächen ... 52
Erstellen einer Schaltfläche ... 53
Kontrollkästchen ... 53
Bewährtes Verfahren: Implementieren von Kontrollkästchen ... 54
Erstellen eines Kontrollkästchens ... 54
Erstellen einer Bitmap ... 56
Erstellen eines benutzerdefinierten Feldes ... 56
Erstellen eines Felds, um Webinhalte anzuzeigen ... 60

Anzeigen von HTML-Inhalt in einem Browserfeld ... 60
Anzeigen von HTML-Inhalt von einer Webseite in einem Browserfeld ... 61
Anzeigen von HTML-Inhalt aus einer Ressource in Ihrer Anwendung ... 63
Konfigurieren eines Browserfeldes ... 65
Senden von Formulardaten an eine Webadresse in einem Browserfeld ... 66
Dialogfelder ... 68
Erstellen eines Dialogfelds ... 69
Dropdown-Listen ... 71
Bewährtes Verfahren: Implementieren von Dropdown-Listen ... 72
Erstellen einer Dropdown-Liste ... 72
Bezeichnungen ... 74
Bewährtes Verfahren: Implementieren von Bezeichnungen ... 75
Erstellen eines Textetiketts ... 75
Listen ... 75
Bewährtes Verfahren: Implementieren von Listen ... 76
Erstellen eines Listenfelds ... 76
Optionsfelder ... 78
Bewährtes Verfahren: Implementieren von Optionsschaltflächen ... 78
Erstellen einer Optionstaste ... 79
Fortschrittsanzeigen ... 81
Bewährtes Verfahren: Implementieren von Fortschrittsanzeigen ... 81
Erstellen einer Fortschrittsanzeige ... 82
Auswahldialogfelder ... 82
Bewährtes Verfahren: Implementieren von Auswahldialogfeldern ... 84
Erstellen einer Datumsauswahl ... 84
Erstellen einer Dateiauswahl ... 87
Suchfelder ... 90
Bewährtes Verfahren: Implementieren von Suchfeldern ... 91
Erstellen eines Suchfelds ... 91
Drehungsfelder ... 93
Bewährtes Verfahren: Implementieren von Drehfeldern ... 94
Erstellen eines Drehfelds ... 95
Textfelder ... 97
Bewährtes Verfahren: Implementieren von Textfeldern ... 98
Erstellen eines Textfelds ... 99

Erstellen eines Datumsfelds ... 100
Erstellen eines Kennwortfelds ... 101
Strukturansichten ... 101
Bewährtes Verfahren: Implementieren von Strukturansichten ... 102
Erstellen eines Feldes für das Anzeigen einer Strukturansicht ... 102

8 Menübefehle ... 104
Erstellen eines Menüs ... 104
Codebeispiel: Erstellen eines Menüs ... 105
Bewährtes Verfahren: Implementieren von Menüs ... 106
Hinzufügen eines Menüeintrags zu einer BlackBerry Device Software-Anwendung ... 107
Hinzufügen eines Menüeintrags zu einer BlackBerry Device Software-Anwendung ... 107
Ändern der Darstellung eines Menüs ... 108
Ändern der Darstellung eines Menüs ... 108
Codebeispiel: Ändern der Darstellung eines Menüs ... 110
Hinzufügen eines Symbols zu einem Menüeintrag ... 111
Hinzufügen eines Symbols zu einem Menüeintrag ... 112
Codebeispiel: Hinzufügen eines Symbols zu einem Menüelement ... 113

9 Benutzerdefinierte Schriftarten ... 115
Installieren und Verwenden einer benutzerdefinierten Schriftart in einer BlackBerry-Java-Anwendung ... 115
Codebeispiel: Installieren und Verwenden einer benutzerdefinierten Schriftart in einer BlackBerry-Java-Anwendung ... 116

10 Rechtschreibprüfung ... 118
Hinzufügen einer Funktion zur Rechtschreibprüfung ... 118
Empfangen von Rechtschreibprüfungsereignissen ... 119

11 Verwandte Ressourcen ... 121

12 Glossar ... 122

13 Rückmeldung ... 123

14 Verlauf der Dokumentrevision ... 124

15 Rechtliche Hinweise ... 128

1 Erstellen einer Benutzeroberfläche, die mit normalen BlackBerry-Benutzeroberflächen kompatibel ist

Mithilfe der Standard-MIDP-APIs und der BlackBerry®-Benutzeroberflächen-APIs können Sie eine Benutzeroberfläche für BlackBerry® Java® Application erstellen.

Die Benutzeroberflächenkomponenten in den BlackBerry-Benutzeroberflächen-APIs wurden dafür entworfen, Layouts und Verhaltensweisen bereitzustellen, die den BlackBerry® Device Software-Anwendungen entsprechen.

- Bildschirmkomponenten können ein Standardbildschirmlayout, ein Standardmenü und ein Standardverhalten bereitstellen, wenn ein BlackBerry-Gerätebenutzer die Esc-Taste drückt, auf das Trackwheel, den Trackball oder das Trackpad klickt oder den Bildschirm berührt.
- Feldkomponenten können die standardmäßigen Benutzeroberflächenelemente für die Datumsauswahl, Optionsschaltflächen, Kontrollkästchen, Listen, Textfelder, Bezeichnungen und Fortschrittsanzeigen bereitstellen.
- Layout-Manager können einer Anwendung die Möglichkeit geben, die Komponenten auf standardmäßige Weise in einem BlackBerry-Gerätebildschirm anzuordnen, z. B. horizontal, vertikal oder von links nach rechts.

Mithilfe der BlackBerry-Benutzeroberflächen-APIs können Sie eine Benutzeroberfläche mit Tabellen, Rastern und anderen speziellen Eigenschaften erstellen. Sie können ein standardmäßiges Java-Ereignismodell verwenden, um bestimmte Arten von Ereignissen zu empfangen und auf sie zu antworten. Eine BlackBerry-Geräteanwendung kann Benutzerereignisse empfangen und auf sie antworten, z. B. wenn der Benutzer auf den Trackball klickt oder Eingaben über die Tastatur vornimmt, und kann Systemereignisse, wie z. B. globale Warnungen, Uhrzeitänderungen und USB-Anschlussverbindungen, empfangen und auf sie antworten.

BlackBerry-Gerät: Eingabe und Navigation für Benutzer 2

BlackBerry®-Geräte umfassen eine Tastatur, ein Trackwheel, Trackpad oder einen Trackball sowie eine Esc-Taste für die Eingabe und die Navigation. Auf BlackBerry-Geräten mit einem SurePress™-Touchscreen entspricht das Klicken auf dem Bildschirm dem Klicken auf den Trackball oder das Trackwheel.

Eine BlackBerry® Java® Application für BlackBerry-Geräte sollte dem nachstehenden Modell für Eingabe und Navigation so weit wie möglich folgen.

- Klicken auf das Trackwheel, Trackpad, den Trackball oder Touchscreen öffnet in der Regel ein Menü.
- Durch Drücken der Esc-Taste können Aktionen abgebrochen werden. Weiterhin ist damit die Rückkehr zum vorherigen Bildschirm möglich. Durch Drücken der Esc-Taste werden Benutzer immer wieder zur Startseite zurückkehren. Durch Halten der Esc-Taste wird der Browser oder die Medienanwendung geschlossen.

Standardmäßig bieten die BlackBerry-Bildschirmobjekte diese Funktion ohne weiteres Anpassen. Sie müssen allerdings die gewünschten Menüelemente und die zusätzliche Benutzeroberflächen- und Navigationslogik hinzufügen.

Touchscreen

Bei BlackBerry®-Geräten mit einem SurePress™-Touchscreen verwenden Benutzer einen Finger, um mit den Anwendungen auf dem Gerät zu interagieren. Durch das Ausführen verschiedener Aktionen auf dem Touchscreen geben Benutzer Text ein und navigieren durch die Bildschirme.

Benutzer können auch Aktionen durch Klicken auf Symbole auf der Shortcut-Leiste oder durch Drücken der Menütaste ausführen.

Bei BlackBerry-Geräten mit einem Touchscreen können Benutzer die folgenden Aktionen ausführen:

Aktion	Ergebnis
Leichtes Berühren des Bildschirms	Diese Aktion markiert ein Element. Wenn Benutzer in einem Textfeld den Bildschirm in der Nähe des Cursors berühren, wird ein umrandetes Feld um den Cursor angezeigt. Dieses Feld hilft Benutzern, den Cursor problemloser zu verschieben.
Tippen auf den Bildschirm	In Anwendungen, die eine Vollbildanzeige unterstützen, wie z. B. BlackBerry® Maps und der BlackBerry® Browser, wird durch diese Aktion die Shortcut-Leiste verborgen bzw. angezeigt..
Zweimaliges Tippen auf den Bildschirm	In einer Webseite, Karte, Bild oder Präsentationsanlage zoomt diese Aktion in die jeweilige Webseite, Karte, Bild oder Präsentationsanlage.

Aktion	Ergebnis
Den Finger auf einem Element halten	Durch diese Aktion wird auf der Shortcut-Leiste ein QuickInfo angezeigt, das die Aktion beschreibt, die das Symbol darstellt. Wenn Benutzer in einer Nachrichtenliste einen Finger auf dem Absender oder den Betreff einer Nachricht halten, sucht das BlackBerry-Gerät nach dem Absender oder dem Betreff.
Berühren und Ziehen eines Elements auf dem Bildschirm	Diese Aktion bewegt den Inhalt auf dem Bildschirm in die entsprechende Richtung. Wenn Benutzer zum Beispiel ein Menüelement berühren und ziehen, bewegt sich die Liste der Menüelemente in die gleiche Richtung. In einem Textfeld bewegt diese Aktion das umrandete Feld und den Cursor in die gleiche Richtung.
Berühren des Bildschirms an zwei Stellen gleichzeitig	Diese Aktion markiert den Text oder die Liste der Elemente, wie z. B. Nachrichten, zwischen den beiden Stellen. Zum Hinzufügen oder Entfernen markierter Texte oder Elemente können Benutzer den Bildschirm an einer anderen Stelle berühren.
Klicken (Drücken) auf den Bildschirm	Leitet eine Aktion ein. Wenn Benutzer zum Beispiel auf ein Element in einer Liste klicken, wird der mit dem Element verknüpfte Bildschirm angezeigt. Diese Aktion entspricht dem Klicken auf das Trackwheel, Trackpad oder den Trackball. In einer Karte, Bild oder Präsentationsanlage zoomt diese Aktion in die jeweilige Karte, Bild oder Präsentationsanlage. In einer Webseite zoomt diese Aktion in die Webseite oder folgt einem Link. In einem Textfeld wird durch diese Aktion der Cursor positioniert. Wenn das Feld Text enthält, erscheint ein umrandetes Feld um den Cursor.
Schnelles Fahren mit einem Finger nach oben oder unten über den Bildschirm	Durch schnelles Fahren mit einem Finger nach oben wird der nächste Bildschirm angezeigt. Durch schnelles Fahren mit einem Finger nach unten wird der vorherige Bildschirm angezeigt. Wenn die Tastatur anzeigt wird, wird durch schnelles Fahren mit einem Finger nach unten die Tastatur verborgen und die Shortcut-Leiste angezeigt.
Schnelles Fahren mit einem Finger nach links oder rechts über den Bildschirm	Diese Aktion zeigt das nächste bzw. vorherige Bild oder Nachricht, oder den nächsten bzw. vorherigen Tag, Woche oder Monat in einem Kalender an.
Schieben mit einem Finger nach oben oder unten über den Bildschirm	In der Kamera, wird durch Schieben eines Fingers ein Objekt herangezoomt. Durch Schieben mit einem Finger nach unten wird aus dem Objekt herausgezoomt.

Aktion	Ergebnis
Schieben eines Fingers in irgendeine Richtung	Diese Aktion führt eine Karte oder Webseite nach. Wenn Benutzer ein Bild heranzoomen, wird durch diese Aktion auch ein Bild nachgeführt.
Drücken der Esc-Taste	Diese Aktion entfernt die Markierung von Text oder einer Liste von Elementen. In einer Webseite, Karte oder einem Bild zoomt diese Aktion um eine Ebene heraus. Benutzer können die Esc-Taste zweimal drücken, um zur ursprünglichen Ansicht zurück zu zoomen.

Trackball oder Trackpad

Bei BlackBerry®-Geräten mit einem Trackball oder Trackpad sind der Trackball oder das Trackpad die primäre Steuerung für die Benutzernavigation. Benutzer können folgende Aktionen ausführen:

- Drehen des Trackballs oder Schieben mit einem Finger auf dem Trackpad, um den Cursor zu bewegen.
- Klicken Sie auf den Trackball oder das Trackpad, um Standardaktionen auszuführen oder ein Kontextmenü zu öffnen.
- Klicken auf den Trackball oder das Trackpad bei gleichzeitigem Drücken der Umschalttaste, um Text zu markieren oder Nachrichten in einer Nachrichtenliste zu markieren.

BlackBerry mit einem Trackball oder Trackpad verfügen auch über eine Menütaste, die sich links vom Trackball bzw. dem Trackpad befindet. Benutzer können die Menütaste drücken, um ein vollständiges Menü verfügbarer Aktionen zu öffnen.

Trackball-Empfindlichkeit

Die Trackball-Empfindlichkeit bezeichnet den Umfang der Trackball-Bewegung, die erforderlich ist, damit das System die Bewegung als Navigationsereignis erkennt und entsprechend ein Navigationsereignis an die Softwareschicht zu senden. Die BlackBerry®-Gerätehardware misst die physische Trackball-Bewegung mithilfe spezieller Einheiten, die als Ticks bezeichnet werden. Wenn die Anzahl der Ticks entlang einer Achse den Schwellenwert des Systems oder einer BlackBerry-Geräteanwendung überschreitet, wird ein Navigationsereignis an die Softwareschicht gesendet, und das System setzt die Tick-Zahl wieder auf Null zurück. Auch nach einer bestimmten Leerlaufzeit wird die Tick-Zahl wieder auf Null zurückgesetzt.

Mit der Trackball-API können Sie die Trackball-Empfindlichkeit festlegen. Eine hohe Trackball-Empfindlichkeit entspricht einem kleineren Tick-Schwellenwert (ein geringe Trackball-Bewegung erzeugt ein Navigationsereignis). Eine niedrige Trackball-Empfindlichkeit entspricht einem größeren Tick-Schwellenwert (ein größere Trackball-Bewegung ist erforderlich, um ein Navigationsereignis zu erzeugen).

Trackball-Bewegung

Mit der Trackball-API filtern Sie die Trackball-Bewegungsdaten, die die Hardware des BlackBerry®-Geräts an die Softwareschicht sendet. Die Trackball-API filtert Bewegungen "Rauschen" oder unerwünschte Bewegungen.

Mit der Trackball-API können Sie auch Einstellungen ändern, z. B. die Beschleunigung beim Bewegen des Trackballs. Eine höhere Einstellung für die Beschleunigung beim Bewegen des Trackballs kann dazu führen, dass die Softwareschicht die Bewegungen als scheinbar schneller erkennt als die BlackBerry-Gerätehardware, solange der Benutzer den Trackball fortlaufend rollt. Die Trackball-Empfindlichkeit steigt vorübergehend an, wenn der Benutzer den Trackball längere Zeit ununterbrochen rollt.

Trackwheel

BlackBerry®-Geräte der BlackBerry® Pearl™ 8100 Series verwenden ein Trackwheel als primäre Steuerung für die Benutzernavigation. Das Trackwheel befindet sich auf der rechten Seite des BlackBerry®-Geräts.

Benutzer können folgende Aktionen ausführen:

- Trackwheel rollen, um den Cursor in vertikaler Richtung zu bewegen
- Trackwheel rollen und gleichzeitig die Alt-Taste drücken, um den Cursor in horizontaler Richtung zu bewegen
- Mit dem Trackwheel klicken, um eine Aktion einzuleiten oder das Menü zu öffnen

Tastatur

Benutzer verwenden die Tastatur hauptsächlich, um Text einzugeben. Buchstabentasten senden dem BlackBerry®-Gerät ein Zeichen. Eine Zusatztaste verändert die Funktion von Zeichentasten. Zu den Zusatztasten gehören die Umschalttaste und die Alt-Taste. Wenn Benutzer eine Zusatztaste drücken, wird eine Texteingabemodusanzeige in der rechten oberen Ecke des Bildschirms angezeigt.

Auf BlackBerry-Geräten ohne Touchscreen können Benutzer die Tastatur auch verwenden, um auf einem Bildschirm zu navigieren (zum Beispiel auf einer Landkarte). Die Navigation mithilfe der Tastatur sollte jedoch immer eine Alternative zur Navigation mithilfe von Trackwheel, Trackball oder Trackpad sein.

BlackBerry-Geräte haben entweder eine komplette Tastatur oder eine reduzierte Tastatur. Auf BlackBerry-Geräten mit einem SurePress™-Touchscreen wird in den meisten Fällen die komplette Tastatur im Querformat und die SureType-Tastatur im Hochformat angezeigt.

Interaktionsmethoden auf BlackBerry-Geräten

BlackBerry-Gerätemodell	Interaktionsmethode
BlackBerry® 7100 Series	Trackwheel
BlackBerry® 8700 Series	Trackwheel
BlackBerry® 8800 Series	Trackball
BlackBerry® Bold™ 9000-Smartphone	Trackball
BlackBerry® Bold™ 9650-Smartphone	Trackpad
BlackBerry® Bold™ 9700-Smartphone	
BlackBerry® Curve™ 8300 Series	Trackball
BlackBerry® Curve™ 8500 Series	Trackpad
BlackBerry® Curve™ 8900-Smartphone	Trackball
BlackBerry® Pearl™ 8100 Series	Trackball
BlackBerry® Pearl™ Flip 8200 Series	Trackball
BlackBerry® Storm™ 9500 Series	Touchscreen
BlackBerry® Tour™ 9630-Smartphone	Trackball

Bildschirme

3

Die Hauptkomponente einer Benutzeroberfläche des BlackBerry®-Geräts ist das `Screen`-Objekt.

Auf einem BlackBerry-Gerät können mehrere Bildschirme gleichzeitig geöffnet sein, aber nur ein Bildschirm zur selben Zeit angezeigt werden. Die BlackBerry® Java® Virtual Machine verwaltet einen sortierten Satz von `Screen`-Objekten in einem Anzeige-Stapel. Der Bildschirm oben im Stapel ist der aktive Bildschirm, der für den BlackBerry-Gerätebenutzer sichtbar ist. Wenn eine BlackBerry-Geräteanwendung einen Bildschirm anzeigt, wird der Bildschirm im Stapel nach oben geschoben. Wenn eine BlackBerry-Geräteanwendung einen Bildschirm schließt, verschwindet der Bildschirm vom Stapel, und der nächste Bildschirm im Stapel wird ausgegeben (und gegebenenfalls neu gezeichnet).

Jeder Bildschirm kann nur einmal im Anzeige-Stapel vorkommen. Die BlackBerry JVM löst einen Laufzeitfehler aus, wenn ein Bildschirm, den die BlackBerry-Geräteanwendung auf den Stapel schiebt, bereits existiert. Eine BlackBerry-Geräteanwendung muss einen Bildschirm vom Anzeige-Stapel nehmen, wenn der Benutzer die Interaktion mit dem Bildschirm beendet, damit die BlackBerry-Geräteanwendung den Speicher effizient einsetzen kann. Sie sollten nur ein paar modale Bildschirme gleichzeitig verwenden, denn jeder Bildschirm verwendet einen separaten Thread.

Die Benutzeroberflächen-APIs initialisieren einfachen `Screen`-Objekte. Nachdem Sie einen Bildschirm erstellen, können Sie Felder und ein Menü hinzufügen, und den Bildschirm dem Benutzer anzeigen, indem Sie ihn im Anzeige-Stapel nach oben schieben. Sie können die Benutzeroberfläche des BlackBerry-Geräts ändern und bei Bedarf neue Feldtypen implementieren. Sie können auch benutzerdefinierte Navigation hinzufügen.

Die `Screen`-Klasse implementiert keine Disambiguierung, die für komplexe Eingabemethoden, wie z. B. internationale Tastaturen, erforderlich ist. Für die Integration der verschiedenen Eingabemethoden können Sie die `Field`-Klasse oder eine ihrer Unterklassen erweitern. Sie sollten `Screen`-Objekte nicht für die Texteingabe verwenden.

Screen-Klassen

Class	Beschreibung
`Bildschirm`	Sie können die `Screen`-Klasse verwenden, um einen Bildschirm zu erstellen, und einen Layout-Manager verwenden, um die Benutzeroberflächen-Komponenten auf dem Bildschirm anzulegen. Sie können eine bestimmte Art von Bildschirm mithilfe der Stile definieren, die von den Konstanten in der übergeordneten `Field`-Klasse definiert werden.

Class	Beschreibung
`FullScreen`	Sie können die `FullScreen`-Klasse verwenden, um einen leeren Bildschirm zu erstellen, dem Sie Benutzeroberflächen-Komponenten in einem standardmäßigen vertikalen Layout hinzufügen können. Wenn Sie einen anderen Layoutstil verwenden wollen, wie z. B. horizontal oder diagonal, können Sie stattdessen die `Screen`-Klasse verwenden und einen Layout-Manager hinzufügen.
`MainScreen`	Sie können die `MainScreen`-Klasse verwenden, um einen Bildschirm mit den folgenden standardmäßigen Benutzeroberflächen-Komponenten zu erstellen: • Standardbildschirmtitel, mit einem `SeparatorField` nach dem Titel • Bildlauffähiger Hauptabschnitt, in einem `VerticalFieldManager` enthalten • Standardmenü mit einem Menüelement "Schließen" • Standardmäßige Schließen-Aktion, die den Bildschirm schließt, wenn der BlackBerry®-Gerätebenutzer auf das Menüelement "Schließen" klickt oder die Esc-Taste drückt Sie sollten den Einsatz eines `MainScreen`-Objekts für den ersten Bildschirm Ihrer BlackBerry-Geräteanwendung in Betracht ziehen, um die Konsistenz mit anderen BlackBerry-Geräteanwendungen beizubehalten.

Verwalten eines Zeichenbereichs

Das `Graphics`-Objekt stellt die gesamte Zeichenoberfläche dar, die für die BlackBerry®-Geräteanwendung verfügbar ist. Um diesen Bereich zu begrenzen, unterteilen Sie ihn in `XYRect`-Objekte. Jedes `XYPoint`-Objekt stellt einen Punkt auf dem Bildschirm dar, der sich aus einer X-Koordinate und einer Y-Koordinate zusammensetzt.

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.ui.Graphics`
   - `net.rim.device.api.ui.XYRect`
   - `net.rim.device.api.ui.XYPoint`
2. Erstellen Sie `XYPoint`-Objekte, um den linken oberen Punkt und den rechten unteren Punkt eines Rechtecks darzustellen. Erstellen Sie mithilfe der `XYPoint`-Objekte ein `XYRect`-Objekt, um einen rechteckigen Clipping-Bereich zu erstellen.

```
XYPoint bottomRight = new XYPoint(50, 50);
XYPoint topLeft = new XYPoint(10, 10);
XYRect rectangle = new XYRect(topLeft, bottomRight);
```

3. Rufen Sie `Graphics.pushContext()` auf, um Zeichenaufrufe durchzuführen, die angeben, dass der Bereichsursprung nicht den Zeichen-Offset anpassen soll. Rufen Sie `Graphics.pushContext()` auf, um den rechteckigen Clipping-Bereich mittels Push zum Kontextstapel zu übertragen. Rufen Sie `Graphics.drawRect()` auf, um ein Rechteck zu zeichnen, und rufen Sie `Graphics.fillRect()` auf, um das Rechteck zu füllen. Rufen Sie `Graphics.popContext()` auf, um den aktuellen Kontext vom Kontextstapel loszulösen.

```
graphics.pushContext(rectangle, 0, 0);
graphics.fillRect(10, 10, 30, 30);
graphics.drawRect(15, 15, 30, 30);
graphics.popContext();
graphics.drawRect(15, 15, 30, 30);
graphics.pushContext(rectangle, 0, 0);
graphics.fillRect(10, 10, 30, 30);
graphics.drawRect(15, 15, 30, 30);
graphics.popContext();
graphics.drawRect(15, 15, 30, 30);
```

4. Rufen Sie `pushRegion()` auf und geben Sie an, dass der Bereichsursprung den Zeichen-Offset anpassen soll. Rufen Sie `Graphics.drawRect()` auf, um ein Rechteck zu zeichnen, und rufen Sie `Graphics.fillRect()` auf, um ein Rechteck zu füllen. Rufen Sie `Graphics.popContext()` auf, um den aktuellen Kontext vom Kontextstapel loszulösen.

```
graphics.pushRegion(rectangle);
graphics.fillRect(10, 10, 30, 30);
graphics.drawRect(15, 15, 30, 30);
graphics.popContext();
```

5. Geben Sie den Teil des `Graphics`-Objekts an, der mittels Push zum Stapel übertragen werden soll.
6. Nachdem Sie `pushContext()` (oder `pushRegion()`) aufgerufen haben, stellen Sie den Teil des `Graphics`-Objekts bereit, der invertiert werden soll.

```
graphics.pushContext(rectangle);
graphics.invert(rectangle);
graphics.popContext();
```

7. Rufen Sie `translate()` auf. Das `XYRect`-Objekt wird von seinem Ursprung (1, 1) in einen Ursprung (20, 20) konvertiert. Nach der Konvertierung reicht der untere Teil des `XYRect`-Objekts über die Grenzen des Grafikkontexts hinaus und schneidet ihn zu.

```
XYRect rectangle = new XYRect(1, 1, 100, 100);
XYPoint newLocation = new XYPoint(20, 20);
rectangle.translate(newLocation);
```

Erstellen eines Bildschirmübergangs

Sie können einen Bildschirmübergang erstellen, um einen visuellen Effekt anzuwenden, der dann erscheint, wenn Ihre Anwendung einen Bildschirm auf einem BlackBerry®-Gerät öffnet oder schließt. Sie können die folgenden Arten von Bildschirmübergängen für Ihre Anwendung mithilfe der `net.rim.device.api.ui.TransitionContext`-Klasse erstellen.

Übergang	Beschreibung
`TRANSITION_FADE`	Dieser Übergang zeigt einen Bildschirm an oder entfernt einen Bildschirm durch allmähliches Ein- oder Ausblenden. Dieser Bildschirmübergang erzeugt den visuellen Effekt des Ein- bzw. Ausblendens durch Änderung der Deckkraft des Bildschirms.
`TRANSITION_SLIDE`	Dieser Übergang zeigt einen Bildschirm an oder entfernt einen Bildschirm, indem dieser auf die Anzeige oder von der Anzeige am Gerät geschoben wird. Sie können Attribute verwenden, um anzugeben, dass sowohl der neue als auch der aktuelle Bildschirm geschoben werden, und dadurch ein Effekt erzeugt wird, als würden sich beide Bildschirme bewegen, oder dass sich der neue über den aktuellen Bildschirm schiebt.
`TRANSITION_WIPE`	Dieser Übergang zeigt einen Bildschirm an oder entfernt einen Bildschirm, indem simuliert wird, dass dieser auf die Anzeige oder von der Anzeige am Gerät gewischt wird.
`TRANSITION_ZOOM`	Dieser Übergang zeigt einen Bildschirm an oder entfernt einen Bildschirm, indem dieser auf die Anzeige oder von der Anzeige am Gerät gezoomt wird.
`TRANSITION_NONE`	Es findet kein Übergang statt.

Jede Art von Bildschirmübergang hat Attribute, die Sie verwenden können, um die visuellen Effekte des Bildschirmübergangs anzupassen. Zum Beispiel können Sie einen schiebenden Effekt so anpassen, dass ein Bildschirm von unten am Bildschirm am Gerät nach oben am Bildschirm geschoben wird. Wenn Sie den Bildschirmübergang nicht anpassen, verwendet die Anwendung die Standardattribute. Weitere Informationen zu den Standardattributen finden Sie in der API-Referenz für das BlackBerry® Java® Development Environment.

Nach der Erstellung eines Bildschirmübergangs müssen Sie diese innerhalb Ihrer Anwendung durch Aufrufen von `UiEngineInstance.setTransition()` registrieren und müssen den zu entfernenden, ausgehenden Bildschirm sowie den anzuzeigenden, eingehende Bildschirm, die Ereignisse, die den Übergang auslösen, und den anzuzeigenden Übergang angeben.

Um eine Beispielanwendung herunterzuladen, die vorführt, wie Bildschirmübergänge verwendet werden, besuchen Sie die Website www.blackberry.com/go/screentransitionssample. Weitere Informationen zu Bildschirmübergängen finden Sie in der API-Referenz für die BlackBerry Java Development Environment.

Anmerkung: Das Thema auf dem BlackBerry-Gerät steuert die visuellen Effekte, die angezeigt werden, wenn ein Benutzer eine Anwendung öffnet. Weitere Informationen finden Sie im *BlackBerry Theme Studio-Benutzerhandbuch*.

Codebeispiel: Erstellen eines Bildschirmübergangs

Das folgende Codebeispiel zeigt einen Übergang durch Verschieben sowie einen Übergang durch allmähliches Ausblenden. Wenn der Benutzer die Anwendung öffnet, wird auf dem BlackBerry®-Gerät der erste Bildschirm angezeigt, der eine Schaltfläche enthält. Wenn der Benutzer auf die Schaltfläche klickt, wird von rechts ein zweiter Bildschirm auf den Gerätebildschirm geschoben. Der zweite Bildschirm wird nach zwei Sekunden automatisch ausgeblendet.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.decor.*;
public class ScreenTransitionSample extends UiApplication implements
    FieldChangeListener {
    private Screen _secondaryScreen;
    private Runnable _popRunnable;
    public static void main(String[] args) {
        ScreenTransitionSample  theApp = new ScreenTransitionSample ();
        theApp.enterEventDispatcher();
    }
    public ScreenTransitionSample () {
        _secondaryScreen = new FullScreen();
        _secondaryScreen.setBackground(
    BackgroundFactory.createSolidBackground(Color.LIGHTBLUE) );
        LabelField labelField = new LabelField("The screen closes automatically in
    two seconds by using a fade transition");
        _secondaryScreen.add(labelField);
        TransitionContext transition = new
    TransitionContext(TransitionContext.TRANSITION_SLIDE);
        transition.setIntAttribute(TransitionContext.ATTR_DURATION, 500);
        transition.setIntAttribute(TransitionContext.ATTR_DIRECTION,
    TransitionContext.DIRECTION_RIGHT);
        transition.setIntAttribute(TransitionContext.ATTR_STYLE,
    TransitionContext.STYLE_PUSH);
        UiEngineInstance engine = Ui.getUiEngineInstance();
        engine.setTransition(null, _secondaryScreen, UiEngineInstance.TRIGGER_PUSH,
    transition);
        transition = new TransitionContext(TransitionContext.TRANSITION_FADE);
        transition.setIntAttribute(TransitionContext.ATTR_DURATION, 500);
        transition.setIntAttribute(TransitionContext.ATTR_KIND,
    TransitionContext.KIND_OUT);
        engine.setTransition(_secondaryScreen, null, UiEngineInstance.TRIGGER_POP,
    transition);
        MainScreen baseScreen = new MainScreen();
        baseScreen.setTitle("Screen Transition Sample");
        ButtonField buttonField = new ButtonField("View Transition",
    ButtonField.CONSUME_CLICK) ;
        buttonField.setChangeListener(this);
        baseScreen.add(buttonField);
        pushScreen(baseScreen);
```

```
        _popRunnable = new Runnable() {
            public void run() {
                popScreen(_secondaryScreen);
            }
        };
    }
    public void fieldChanged(Field field, int context)
    {
        pushScreen(_secondaryScreen);
        invokeLater(_popRunnable, 2000, false);
    }
}
```

Festlegen des Formats und der Richtung des Bildschirms

In Touchscreen-Anwendungen können Sie sowohl das Format als auch die Richtung des Bildschirms berücksichtigen. Das Format bezieht sich auf das Seitenverhältnis des Bildschirms. Die Richtung bezieht sich auf den Zeichenbereich des Bildschirms.

Format

Der Benutzer eines BlackBerry®-Geräts mit Touchscreen kann das Format des Bildschirms durch Drehen des Geräts ändern. Wenn ein Benutzer das BlackBerry-Gerät so hält, dass sich das BlackBerry-Logo oben befindet, weist der Bildschirm das Hochformat auf. Wenn der Benutzer das Gerät um 90 Grad nach links oder rechts dreht, weist der Bildschirm das Querformat auf.

Sie können mithilfe der `net.rim.device.api.system.Display`-Klasse das Format des Bildschirms abrufen. Die `Display`-Klasse enthält die Konstanten, die den Formaten entsprechen, die das Gerät mit Touchscreen zum Anzeigen von Informationen verwenden kann. Das Hochformat entspricht beispielsweise der Konstante `Display.ORIENTATION_PORTRAIT`, das Querformat der Konstante `Display.ORIENTATION_LANDSCAPE` und das quadratische Format der Konstante `Display.ORIENTATION_SQUARE`.

Um das Format des Bildschirms abzurufen, können Sie die `Display.getOrientation()`-Methode aufrufen. Diese Methode gibt eine Ganzzahl zurück, die einem der Formate entspricht, die auf dem BlackBerry-Gerät verwendet werden können.

Damit Formatänderungen erkannt werden, können Sie `Screen.sublayout()` überschreiben. Rufen Sie in dieser Methode `super.sublayout()` auf und achten Sie auf Änderungen der `width`- und `height`-Werte.

Richtung

Ein BlackBerry-Gerät mit Touchscreen kann Informationen auf dem Bildschirm in verschiedenen Richtungen anzeigen. Die Richtung bezieht sich auf das obere Ende des Zeichenbereichs des Bildschirms, das zur Position des BlackBerry-Logos relativ ist. Die Richtung ist "Norden", wenn das obere Ende des Zeichenbereichs die Bildschirmseite ist, die dem BlackBerry-Logo am nächsten liegt, "Westen", wenn sich das obere Ende des Zeichenbereichs links vom BlackBerry-Logo befindet, und "Osten", wenn sich das obere Ende des Zeichenbereichs rechts vom BlackBerry-Logo befindet.

Mithilfe der `net.rim.device.api.system.Display`-Klasse, der `net.rim.device.api.ui.Ui`-Klasse und der `net.rim.device.api.ui.UiEngineInstance`-Klasse können Sie die Richtung steuern, die das BlackBerry-Gerät zum Anzeigen von Informationen auf dem Bildschirm verwendet. Die `Display`-Klasse enthält Konstanten, die den Richtungen

entsprechen, die das Gerät zum Anzeigen von Informationen verwenden kann. Norden entspricht beispielsweise der Konstante `Display.DIRECTION_NORTH`, Westen der Konstante `Display.DIRECTION_WEST` und Osten der Konstante `Display.DIRECTION_EAST`.

Sie können ein Objekt der `net.rim.device.api.ui.Ui`-Klasse erstellen und `Ui.getUiEngineInstance()` aufrufen, um ein Objekt der `UiEngineInstance`-Klasse abzurufen. Durch Aufrufen von `UiEngineInstance.setAcceptableDirections()` mit einer der Konstanten, die Richtungen aus der `Display`-Klasse entsprechen, wird die Richtung festgelegt, die das BlackBerry-Gerät zum Anzeigen von Informationen verwenden kann.

Codebeispiel: Abrufen der Bildschirmausrichtung

```
switch(Display.getOrientation())
{
   case Display.ORIENTATION_LANDSCAPE:
      Dialog.alert("Screen orientation is landscape"); break;
   case Display.ORIENTATION_PORTRAIT:
      Dialog.alert("Screen orientation is portrait"); break;
   case Display.ORIENTATION_SQUARE:
      Dialog.alert("Screen orientation is square"); break;
   default:
      Dialog.alert("Screen orientation is not known"); break;
}
```

Codebeispiel: Erzwingen des Hochformats in einer BlackBerry-API-Anwendung

```
// Use code like this before invoking UiApplication.pushScreen()
int direction = Display.DIRECTION_NORTH;
Ui.getUiEngineInstance().setAcceptableDirections(direction);
```

Codebeispiel: Erzwingen des Querformats in einer MIDlet-Anwendung

```
// Use code like this before invoking Display.setCurrent() in the MIDlet constructor
DirectionControl dc =
  (DirectionControl) ((Controllable) Display.getDisplay(this)).
    getControl("net.rim.device.api.lcdui.control.DirectionControl");
int directions = DirectionControl.DIRECTION_EAST | DirectionControl.DIRECTION_WEST;
dc.setAcceptableScreenDirections(directions);
```

Abrufen der Ausrichtung des Touchscreen

1. Importieren Sie die `net.rim.device.api.system.Display`-Klasse.
2. Rufen Sie `net.rim.device.api.system.Display.getOrientation()` auf.

```
int orientation = Display.getOrientation();
```

Schränken Sie die Richtung des Touchscreens ein

1. Importieren Sie die folgenden Klassen:

- `net.rim.device.api.ui.Ui`
- `net.rim.device.api.ui.UiEngineInstance`

2. Rufen Sie `net.rim.device.api.ui.Ui.getUiEngineInstance()` auf.

```
UiEngineInstance _ue;
_ue = Ui.getUiEngineInstance();
```

3. Rufen Sie `net.rim.device.api.ui.UiEngineInstance.setAcceptableDirections(byte flags)` auf und übergeben Sie das Argument für die Richtung des `Bildschirms`.

```
_ue.setAcceptableDirections(Display.DIRECTION_WEST);
```

Benachrichtigen lassen, wenn sich die Größe des Zeichenbereichs auf dem Touchscreen ändert

1. Importieren Sie die folgenden Klassen:
   - `javax.microedition.lcdui.Canvas`
   - `net.rim.device.api.ui.component.Dialog`
2. Erweitern Sie `javax.microedition.lcdui.Canvas`.
3. Überschreiben Sie `Canvas.sizeChanged(int, int)`.

```
protected void sizeChanged(int w, int h) {
   Dialog.alert("The size of the Canvas has changed");
}
```

Beschleunigungssensor

4

Ein BlackBerry®-Gerät mit einem Touchscreen enthält einen Beschleunigungssensor, der die Aufgabe hat, die Ausrichtung und die Beschleunigung des BlackBerry-Geräts wahrzunehmen. Wenn ein BlackBerry-Gerätebenutzer das BlackBerry-Gerät bewegt, kann der Beschleunigungssensor die Bewegung im 3D-Bereich entlang der X-, Y- und Z-Achse wahrnehmen. Ein Gerätebenutzer kann die Ausrichtung vom Gerät ändern, das die Bildschirmrichtung von einem Bildschirm für eine BlackBerry-Geräteanwendung zwischen Hoch- und Querformat ändern kann.

Sie können die Accelerometer APIs im `net.rim.device.api.system`-Paket für die Reaktion auf die Ausrichtung und Beschleunigung des BlackBerry-Geräts verwenden. Zum Beispiel können Sie eine Spieleanwendung ändern, um die Richtung und die Geschwindigkeit von einem bewegenden Objekt auf dem Bildschirm zu ändern, während ein Benutzer das BlackBerry-Gerät mit verschiedenen Geschwindigkeiten bewegt und dreht.

Um eine Beispielanwendung herunterzuladen, die vorführt, wie Bildschirmübergänge verwendet werden, besuchen Sie www.blackberry.com/go/accelerometersample. Weitere Informationen zur Beispielanwendung finden Sie unter www.blackberry.com/go/devguides in folgenden Dokumenten: *Accelerometer Sample Application Overview (Überblick über die Beispielanwendung Beschleunigungssensor)*.

Arten von Beschleunigungssensor-Daten

Eine BlackBerry®-Geräteanwendung kann Daten vom Beschleunigungssensor abrufen.

Datentyp	Beschreibung
Ausrichtung	Die Ausrichtung des BlackBerry-Geräts in Bezug auf den Boden.
Beschleunigung	Die Drehbeschleunigung des BlackBerry-Geräts.

Weitere Informationen zu Datentypen des Beschleunigungssensors finden Sie in der API-Referenz für das BlackBerry® Java® Development Environment.

Abrufen von Beschleunigungssensordaten

Der Beschleunigungssensor dient dazu, die Richtung und Geschwindigkeit der Bewegung entlang einer X-, Y- und Z-Achse nachzuverfolgen, wenn ein BlackBerry®-Gerätebenutzer das BlackBerry-Gerät bewegt. Die X-Achse ist zur Breite des BlackBerry-Geräts parallel. Die Y-Achse ist zur Länge des BlackBerry-Geräts parallel. Die Z-Achse ist zur Tiefe (von der Vorder- bis zur Rückseite) des BlackBerry-Geräts parallel. Weitere Informationen zur X-, Y- und Z-Achse des Beschleunigungssensors finden Sie in der Beschreibung der `net.rim.device.api.system.AccelerometerSensor`-Klasse in der API-Referenz für BlackBerry® Java® Development Environment.

Sie können einer BlackBerry-Geräteanwendung ermöglichen, auf die Beschleunigung eines BlackBerry-Geräts zu reagieren, das über einen Beschleunigungssensor verfügt. Ein BlackBerry-Gerätebenutzer könnte beispielsweise das BlackBerry-Gerät bewegen, um die Richtung und Geschwindigkeit eines Objekts zu steuern, das sich in einer Spieleanwendung durch ein Labyrinth bewegt.

Sie können die Accelerometer APIs im `net.rim.device.api.system`-Paket für die Reaktion auf die Beschleunigung des BlackBerry-Geräts verwenden. Sie müssen zuerst feststellen, ob das BlackBerry-Gerät einen Beschleunigungssensor unterstützt, indem Sie `net.rim.device.api.system.AccelerometerSensor.isSupported()` aufrufen. Wenn die Methode den Wert `true` zurückgibt, unterstützt das BlackBerry-Gerät einen Beschleunigungssensor.

Mithilfe der `AccelerometerData`-Klasse können Sie die Richtung identifizieren, in der der Benutzer das BlackBerry-Gerät bewegt. Durch Aufrufen von `AccelerometerData.getOrientation()` wird eine der Konstanten der `AccelerometerSensor`-Klasse zurückgegeben, die der Richtung des BlackBerry-Geräts entsprechen. Wenn `AccelerometerData.getOrientation()` beispielsweise einen Wert gleich `AccelerometerSensor.ORIENTATION_LEFT_UP` zurückgibt, weist die linke Seite des BlackBerry-Geräts nach oben.

Mithilfe der `AccelerometerSensor`-Klasse können Sie Beschleunigungsdaten vom BlackBerry-Gerät abrufen. Durch Aufrufen von `AccelerometerSensor.openRawDataChannel()` wird ein Objekt der `net.rim.device.api.system.AccelerometerSensor.Channel`-Klasse zurückgegeben. Die `AccelerometerSensor.Channel`-Klasse ermöglicht es Ihnen, eine Verbindung zum Beschleunigungssensor herzustellen. Sie können Daten vom Beschleunigungssensor abrufen, indem Sie `AccelerometerSensor.Channel.getLastAccelerationData()` aufrufen.

Durch die Aufrechterhaltung einer Verbindung zum Beschleunigungssensor wird der Akku des BlackBerry-Geräts beansprucht. Wenn die BlackBerry-Geräteanwendung keine Daten mehr vom Beschleunigungssensor abrufen muss, sollten Sie `AccelerometerSensor.Channel.close()` aufrufen, um die Verbindung zu schließen.

Codebeispiel: Abrufen von Daten vom Beschleunigungssensor

```
short[] xyz = new short[3];
while( running ) {
    channel.getLastAccelerationData(xyz);
}
channel.close();
```

Abrufen der Beschleunigungssensordaten in bestimmten Abständen

Wenn eine BlackBerry®-Geräteanwendung einen Kanal für den Beschleunigungssensor öffnet, während sich die Anwendung im Vordergrund befindet, und die Anwendung sich anschließend im Hintergrund befindet, pausiert der Kanal und der Beschleunigungssensor wird nicht abgefragt. Wenn eine BlackBerry-Geräteanwendung `AccelerometerSensor.Channel. getLastAccelerationData(short[])` in kurzen Abständen aufruft, oder wenn das BlackBerry-Gerät nicht bewegt wird, können durch die Methode doppelte Werte zurückgegeben werden.

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.system.AccelerometerSensor.Channel;`

- `net.rim.device.api.system.AccelerometerSensor`

2. Öffnen Sie einen Kanal für den Beschleunigungssensor. Während ein Kanal offen ist, wird von der BlackBerry-Geräteanwendung der Beschleunigungssensor nach Informationen abgefragt.

```
Channel rawDataChannel = AccelerometerSensor.openRawDataChannel
( Application.getApplication() );
```

3. Erstellen Sie ein Array zum Speichern der Beschleunigungssensordaten.

```
short[] xyz = new short[ 3 ];
```

4. Erstellen Sie einen Thread.

```
 while( running ) {
```

5. Rufen Sie `Channel.getLastAccelerationData(short[])` auf, um Daten vom Beschleunigungssensor abzurufen.

```
rawDataChannel.getLastAccelerationData( xyz );
```

6. Rufen Sie `Thread.sleep()` auf, um den Abstand zwischen Abfragen zum Beschleunigungssensor in Millisekunden anzugeben.

```
Thread.sleep( 500 );
```

7. Rufen Sie `Channel.close()` auf, um den Kanal zum Beschleunigungssensor zu schließen.

```
rawDataChannel.close();
```

Abfragen des Beschleunigungssensors, wenn sich die Anwendung im Vordergrund befindet

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.system.AccelerometerChannelConfig`
   - `net.rim.device.api.system.AccelerometerSensor.Channel`
2. Öffnen Sie einen Kanal, um Beschleunigungsdaten vom Beschleunigungssensor abzurufen.

```
Channel channel = AccelerometerSensor.openRawAccelerationChannel
( Application.getApplication());
```

3. Rufen Sie `Thread.sleep()` auf, um den Abstand zwischen Abfragen zum Beschleunigungssensor in Millisekunden anzugeben.

```
short[] xyz = new short[ 3 ];
  while( running ) {
       channel.getLastAccelerationData( xyz );
       Thread.sleep( 500 );
  }
```

4. Rufen Sie `Channel.close()` auf, um den Kanal zum Beschleunigungssensor zu schließen.

```
channel.close();
```

Abfragen des Beschleunigungssensors, wenn sich die Anwendung im Hintergrund befindet

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.system.AccelerometerChannelConfig`
   - `net.rim.device.api.system.AccelerometerSensor.Channel;`
2. Erstellen Sie eine Konfiguration für einen Kanal zum Beschleunigungssensor.

```
AccelerometerChannelConfig channelConfig = new AccelerometerChannelConfig
( AccelerometerChannelConfig.TYPE_RAW );
```

3. Rufen `Sie AccelerometerChannelConfig.setBackgroundMode(Boolean)` auf, um den Support für eine Anwendung anzugeben, die sich im Hintergrund befindet.

```
channelConfig.setBackgroundMode( true );
```

4. Rufen Sie `AccelerometerSensor.openChannel()` auf, um einen Kanal, der sich im Hintergrund befindet, zum Beschleunigungssensor zu öffnen.

```
Channel channel = AccelerometerSensor.openChannel( Application.getApplication(),
channelConfig );
```

5. Rufen Sie `Thread.sleep()` auf, um den Abstand zwischen Abfragen zum Beschleunigungssensor in Millisekunden anzugeben.

```
short[] xyz = new short[ 3 ];
  while( running ) {
       channel.getLastAccelerationData( xyz );
       Thread.sleep( 500 );
}
```

6. Rufen Sie `Channel.close()` auf, um den Kanal zum Beschleunigungssensor zu schließen.

```
channel.close();
```

Speichern der ausgelesenen Beschleunigungssensorwerte in einem Puffer

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.system.AccelerometerChannelConfig`
   - `net.rim.device.api.system.AccelerometerSensor.Channel;`
2. Erstellen Sie eine Konfiguration für einen Kanal zum Beschleunigungssensor.

```
AccelerometerChannelConfig channelConfig = new AccelerometerChannelConfig
( AccelerometerChannelConfig.TYPE_RAW );
```

3. Rufen Sie `AccelerometerChannelConfig.setSamplesCount()` auf, um die Anzahl der im Puffer zu speichernden ausgelesenen Beschleunigungswerte anzugeben. Jedes Element im Puffer beinhaltet die ausgelesenen Beschleunigungswerte für die X-, Y- und Z-Achse sowie Daten über den Zeitpunkt des Auslesens.

```
channelConfig.setSamplesCount( 500 );
```

4. Rufen Sie `AccelerometerSensor.openChannel()` auf, einen Kanal zum Beschleunigungssensor zu öffnen.

```
Channel bufferedChannel = AccelerometerSensor.openChannel
( Application.getApplication(), channelConfig );
```

Abrufen von Beschleunigungssensordaten aus einem Puffer

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.system.AccelerometerData;`
   - `net.rim.device.api.system.AccelerometerSensor.Channel;`
2. Fragen Sie Beschleunigungssensordaten aus dem Puffer ab.

```
AccelerometerData accData = bufferedChannel.getAccelerometerData();
```

3. Rufen Sie `AccelerometerData.getNewBatchLength()` auf, um die Anzahl der Werte zu erhalten, die seit der letzten Abfrage abgerufen wurden.

```
int newBatchSize = accData.getNewBatchLength();
```

4. Rufen Sie `AccelerometerData.getXAccHistory()`, `AccelerometerData.getYAccHistory()` und `AccelerometerData.getZAccHistory()` auf, um für jede Achse Beschleunigungssensordaten aus dem Puffer abzurufen.

```
short[] xAccel = accData.getXAccHistory();
short[] yAccel = accData.getYAccHistory();
short[] zAccel = accData.getZAccHistory();
```

Ermitteln des Zeitpunkts des Abrufens eines Werts vom Beschleunigungssensor

1. Importieren Sie die `net.rim.device.api.system.AccelerometerData`-Klasse.
2. Fragen Sie Beschleunigungssensordaten aus dem Puffer ab.

```
AccelerometerData accData;
accData = bufferedChannel.getAccelerometerData();
```

3. Rufen Sie `AccelerometerData.getSampleTsHistory()` auf.

```
long[] queryTimestamps = accData.getSampleTsHistory();
```

Ereignisse

5

Reagierenauf Navigationsereignisse

Sie können die `Screen`-Navigationsmethoden verwenden, um eine BlackBerry®-Geräteanwendung zu erstellen. Wenn Ihre vorhandene BlackBerry-Geräteanwendung die `TrackwheelListener`-Schnittstelle implementiert, aktualisieren Sie Ihre BlackBerry-Geräteanwendung, um die `Screen`-Navigationsmethoden zu verwenden.

1. Importieren Sie die `net.rim.device.api.ui.Screen`-Klasse.
2. Verwalten Sie Navigationsereignisse durch Erweitern der `net.rim.device.api.ui.Screen`-Klasse (oder einer der Unterklassen) und Überschreiben der folgenden Navigationsmethoden:

```
navigationClick(int status, int time)
navigationUnclick(int status, int time)
navigationMovement(int dx, int dy, int status, int time)
```

Bestimmen der Art der Eingabemethode

1. Importieren Sie eine oder mehrere der folgenden Klassen:
   - `net.rim.device.api.ui.Screen`
   - `net.rim.device.api.ui.Field`
2. Importieren Sie die `net.rim.device.api.system.KeypadListener`-Schnittstelle.
3. Implementieren Sie die `net.rim.device.api.system.KeypadListener`-Schnittstelle.
4. Erweitern Sie die `Screen`-Klasse, die `Field`-Klasse oder beide.
5. In Ihrer Implementierung der Methoden `navigationClick`, `navigationUnclick` oder `navigationMovement` führen Sie einen bitweisen AND-Vorgang auf dem `status`-Parameter aus, um mehr Informationen zum Ereignis zu erzeugen. Implementieren Sie die `navigationClick(int status, int time)`-Methode, um zu bestimmen, ob das Trackwheel oder ein Vier-Wege-Navigationseingabegerät (zum Beispiel ein Trackball) ein Ereignis auslöst.

```
public boolean navigationClick(int status, int time) {
    if ((status & KeypadListener.STATUS_TRACKWHEEL) ==
KeypadListener.STATUS_TRACKWHEEL) {
        //Input came from the trackwheel
    } else if ((status & KeypadListener.STATUS_FOUR_WAY) ==
KeypadListener.STATUS_FOUR_WAY) {
        //Input came from a four way navigation input device
    }
    return super.navigationClick(status, time);
}
```

Weitere Informationen zur `net.rim.device.api.lbs.KeypadListener`-Klasse finden Sie in der API-Referenz für BlackBerry® Java® Development Environment.

Reagieren auf Touchscreen-Ereignisse

BlackBerry®-Gerätebenutzer können auf einem Gerät mit Touchscreen die gleichen Aktionen wie auf einem BlackBerry-Gerät mit Trackball ausführen. Beispielsweise können BlackBerry-Gerätebenutzer den Touchscreen verwenden, um ein Menü zu öffnen, durch eine Liste von Optionen zu scrollen und eine Option auszuwählen.

Wenn Sie mithilfe einer älteren Version als Version 4.7 von BlackBerry® Java® Development Environment eine BlackBerry-Geräteanwendung erstellen, können Sie die JAD-Datei der Anwendung ändern, damit die Anwendung reagieren kann, wenn ein BlackBerry-Gerätebenutzer den Touchscreen berührt. In der JAD-Datei legen Sie die Option `RIM-TouchCompatibilityMode` auf `false` fest.

Wenn Sie mit BlackBerry JDE Version 4.7 oder höher eine BlackBerry-Geräteanwendung erstellen, können Sie der Anwendung die Erkennung der Aktion ermöglichen, die der BlackBerry-Gerätebenutzer auf dem Touchscreen ausführt, indem Sie die `net.rim.device.api.ui.Screen`-Klasse, die `net.rim.device.api.ui.Field`-Klasse oder eine der Unterklassen der `Field`-Klasse erweitern und die `touchEvent()`-Methode überschreiben. Vergleichen Sie innerhalb der `touchEvent()`-Methode den Wert, den `TouchEvent.getEvent()` zurückgibt, mit den Konstanten aus der `net.rim.device.api.ui.TouchEvent`-Klasse und der `net.rim.device.api.ui.TouchGesture`-Klasse.

Die `TouchEvent`-Klasse enthält die Konstanten, die die verschiedenen Aktionen darstellen, die ein Benutzer auf dem Touchscreen ausführen kann. Die Aktion "Klicken" entspricht beispielsweise der Konstante `TouchEvent.CLICK`, die Aktion "Berühren" der Konstante `TouchEvent.DOWN` und die Aktion "Verschieben" der Konstante `TouchEvent.MOVE`.

Die `TouchGesture`-Klasse enthält die Konstanten, die die verschiedenen Gesten darstellen, die ein Benutzer auf einem Touchscreen ausführen kann. Die Geste "Tippen" entspricht beispielsweise der Konstante `TouchGesture.TAP` und die Geste "Fingerstrich" der Konstante `TouchGesture.SWIPE`.

Codebeispiel: Identifizieren der Aktion, die ein Benutzer auf dem Touchscreen ausführt

Im folgenden Codebeispiel wird eine `switch`-Anweisung zur Identifizierung der Aktion verwendet.

```
protected boolean touchEvent(TouchEvent message) {
switch(message.getEvent()) {
    case TouchEvent.CLICK:
    Dialog.alert("A click action occurred");
    return true;
case TouchEvent.DOWN:
    Dialog.alert("A down action occurred");
    return true;
case TouchEvent.MOVE:
    Dialog.alert("A move action occurred");
    return true;
}
return false;
}
```

Antworten auf Systemereignisse, während der Benutzer den Bildschirm berührt

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.ui.TouchEvent`
   - `net.rim.device.api.ui.Field`
   - `net.rim.device.api.ui.Manager`
   - `net.rim.device.api.ui.Screen`
   - `net.rim.device.api.ui.component.Dialog`
2. Erstellen Sie eine Klasse, die die `Manager`-Klasse, die `Screen`-Klasse, die `Field`-Klasse oder eine der `Field`-Unterklassen erweitert.

   ```
   public class newButtonField extends ButtonField {
   ```
3. Rufen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode `TouchEvent.getEvent()` auf.
4. Prüfen Sie, ob der von `TouchEvent.getEvent()` zurückgegebene Wert gleich `TouchEvent.CANCEL` ist.

   ```
   protected boolean touchEvent(TouchEvent message) {
   switch(message.getEvent()) {
     case TouchEvent.CANCEL:
      Dialog.alert("System event occurred while processing touch events");
       return true;
      }
    return false;
   }
   ```

Reagieren auf einen Benutzer, der einen Finger schnell nach oben über den Bildschirm bewegt

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.ui.TouchEvent`
   - `net.rim.device.api.ui.TouchGesture`
   - `net.rim.device.api.ui.Field`
   - `net.rim.device.api.ui.Manager`
   - `net.rim.device.api.ui.Screen`
   - `net.rim.device.api.ui.component.Dialog`
2. Erstellen Sie eine Klasse, die die `Manager`-Klasse, die `Screen`-Klasse, die `Field`-Klasse, oder eine der `Field`-Unterklassen erweitert.

```
public class newButtonField extends ButtonField {
```

3. Rufen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode `TouchEvent.getEvent()` auf.
4. Prüfen Sie, ob der von `TouchGesture.getSwipeDirection()` zurückgegebene Wert gleich `TouchGesture.SWIPE_NORTH` ist.

```
protected boolean touchEvent(TouchEvent message) {
   switch(message.getEvent()) {
      case TouchEvent.GESTURE:
         TouchGesture gesture = message.getGesture();
         switch(gesture.getEvent()) {
            case TouchGesture.SWIPE:
               if(gesture.getSwipeDirection() == TouchGesture.SWIPE_NORTH) {
                  Dialog.alert("Upward swipe occurred");
                  return true;
               }
         }
         return false;
   }
}
```

Reagieren auf einen Benutzer, der einen Finger schnell nach unten über den Bildschirm bewegt

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.ui.TouchEvent`
   - `net.rim.device.api.ui.TouchGesture`
   - `net.rim.device.api.ui.Field`
   - `net.rim.device.api.ui.Manager`
   - `net.rim.device.api.ui.Screen`
   - `net.rim.device.api.ui.component.Dialog`
2. Erstellen Sie eine Klasse, die die `Manager`-Klasse, die `Screen`-Klasse, die `Field`-Klasse, oder eine der `Field`-Unterklassen erweitert.

```
public class newButtonField extends ButtonField {
```

3. Rufen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode `TouchEvent.getEvent()` auf.
4. Prüfen Sie, ob der von `TouchGesture.getSwipeDirection()` zurückgegebene Wert gleich `TouchGesture.SWIPE_SOUTH` ist.

```
protected boolean touchEvent(TouchEvent message) {
   switch(message.getEvent()) {
   case TouchEvent.GESTURE:
      TouchGesture gesture = message.getGesture();
      switch(gesture.getEvent()) {
         case TouchGesture.SWIPE:
            if(gesture.getSwipeDirection() == TouchGesture.SWIPE_SOUTH) {
               Dialog.alert("Downward swipe occurred");
               return true;
            }
      }
      return false;
   }
}
```

Reagieren auf einen Benutzer, der einen Finger schnell nach rechts über den Bildschirm bewegt

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.ui.TouchEvent`
   - `net.rim.device.api.ui.TouchGesture`
   - `net.rim.device.api.ui.Field`
   - `net.rim.device.api.ui.Manager`
   - `net.rim.device.api.ui.Screen`
   - `net.rim.device.api.ui.component.Dialog`
2. Erstellen Sie eine Klasse, die die `Manager`-Klasse, die `Screen`-Klasse, die `Field`-Klasse oder eine der `Field`-Unterklassen erweitert.

   ```
   public class newButtonField extends ButtonField {
   ```
3. Rufen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode `TouchEvent.getEvent()` auf.
4. Prüfen Sie, ob der von `TouchGesture.getSwipeDirection()` zurückgegebene Wert gleich `TouchGesture.SWIPE_EAST` ist.

   ```
   protected boolean touchEvent(TouchEvent message) {
      switch(message.getEvent()) {
         case TouchEvent.GESTURE:
            TouchGesture gesture = message.getGesture();
            switch(gesture.getEvent()) {
               case TouchGesture.SWIPE:
                  if(gesture.getSwipeDirection() == TouchGesture.SWIPE_EAST) {
                     Dialog.alert("Eastward swipe occurred");
                     return true;
                  }
   ```

```
            }
            return false;
        }
}
```

Reagieren auf einen Benutzer, der schnell einen Finger nach rechts über den Bildschirm bewegt

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.ui.TouchEvent`
   - `net.rim.device.api.ui.TouchGesture`
   - `net.rim.device.api.ui.Field`
   - `net.rim.device.api.ui.Manager`
   - `net.rim.device.api.ui.Screen`
   - `net.rim.device.api.ui.component.Dialog`
2. Erstellen Sie eine Klasse, die die `Manager`-Klasse, die `Screen`-Klasse, die `Field`-Klasse oder eine der `Field`-Unterklassen erweitert.

   ```
   public class newButtonField extends ButtonField {
   ```
3. Rufen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode `TouchEvent.getEvent()` auf.
4. Prüfen Sie, ob der von `TouchGesture.getSwipeDirection()` zurückgegebene Wert gleich `TouchGesture.SWIPE_WEST` ist.

```
protected boolean touchEvent(TouchEvent message) {
   switch(message.getEvent()) {
      case TouchEvent.GESTURE:
         TouchGesture gesture = message.getGesture();
         switch(gesture.getEvent()) {
            case TouchGesture.SWIPE:
               if(gesture.getSwipeDirection() == TouchGesture.SWIPE_WEST) {
                  Dialog.alert("Westward swipe occurred");
                  return true;
               }
         }
         return false;
   }
}
```

Reagieren auf einen Benutzer, der auf den Bildschirm klickt

1. Importieren Sie die folgenden Klassen:

- `net.rim.device.api.ui.TouchEvent`
- `net.rim.device.api.ui.Field`
- `net.rim.device.api.ui.Manager`
- `net.rim.device.api.ui.Screen`
- `net.rim.device.api.ui.component.Dialog`

2. Erstellen Sie eine Klasse, die die `Manager`-Klasse, die `Screen`-Klasse, die `Field`-Klasse, oder eine der `Field`-Unterklassen erweitert.

```
public class newButtonField extends ButtonField {
```

3. Rufen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode `TouchEvent.getEvent()` auf.
4. Prüfen Sie, ob der von `TouchEvent.getEvent()` zurückgegebene Wert gleich `TouchEvent.CLICK` ist.

```
protected boolean touchEvent(TouchEvent message) {
   switch(message.getEvent()) {
      case TouchEvent.CLICK:
         Dialog.alert("CLICK occurred");
         return true;
   }
   return false;
}
```

Reagieren auf einen Benutzer, der den Bildschirm zweimal schnell berührt

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.ui.TouchEvent`
   - `net.rim.device.api.ui.TouchGesture`
   - `net.rim.device.api.ui.Field`
   - `net.rim.device.api.ui.Manager`
   - `net.rim.device.api.ui.Screen`
   - `net.rim.device.api.ui.component.Dialog`
2. Erstellen Sie eine Klasse, die die `Manager`-Klasse, die `Screen`-Klasse, die `Field`-Klasse oder eine der `Field`-Unterklassen erweitert.

```
public class newButtonField extends ButtonField {
```

3. Prüfen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode das Auftreten eines `TouchGesture.TAP`-Ereignisses, und dass `TouchGesture.getTapCount` den Wert 2 zurückgibt.

```
protected boolean touchEvent(TouchEvent message) {
   switch(message.getEvent()) {
      case TouchEvent.GESTURE:
         TouchGesture gesture = message.getGesture();
         switch(gesture.getEvent()) {
```

```
            case TouchGesture.TAP:
               if(gesture.getTapCount() == 2) {
                  Dialog.alert("Double tap occurred");
                  return true;
               }
         }
   }
 return false;
}
```

Reagieren auf einen Benutzer, der ein Element auf dem Bildschirm berührt und zieht

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.ui.TouchEvent`
   - `net.rim.device.api.ui.Field`
   - `net.rim.device.api.ui.Manager`
   - `net.rim.device.api.ui.Screen`
   - `net.rim.device.api.ui.component.Dialog`
2. Erstellen Sie eine Klasse, die die `Manager`-Klasse, die `Screen`-Klasse, die `Field`-Klasse, oder eine der `Field`-Unterklassen erweitert.

   ```
   public class newButtonField extends ButtonField {
   ```
3. Rufen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode `TouchEvent.getEvent()` auf.
4. Prüfen Sie, ob der von `TouchEvent.getEvent()` zurückgegebene Wert gleich `TouchEvent.MOVE` ist.

```
protected boolean touchEvent(TouchEvent message) {
   switch(message.getEvent()) {
      case TouchEvent.MOVE:
         Dialog.alert("Move event occurred");
         return true;
   }
   return false;
}
```

Reagieren auf einen Benutzer, der den Bildschirm leicht berührt

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.ui.TouchEvent`
   - `net.rim.device.api.ui.TouchGesture`
   - `net.rim.device.api.ui.Field`

- `net.rim.device.api.ui.Manager`
- `net.rim.device.api.ui.Screen`
- `net.rim.device.api.ui.component.Dialog`

2. Erstellen Sie eine Klasse, die die `Manager`-Klasse, die `Screen`-Klasse, die `Field`-Klasse, oder eine der `Field`-Unterklassen erweitert.

```
public class newButtonField extends ButtonField {
```

3. Rufen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode `TouchEvent.getEvent()` auf.
4. Prüfen Sie, ob der von `TouchEvent.getEvent()` zurückgegebene Wert gleich `TouchEvent.DOWN` ist.

```
protected boolean touchEvent(TouchEvent message) {
   switch(message.getEvent()) {
      case TouchEvent.DOWN:
         Dialog.alert("Touch occurred");
         return true;
   }
   return false;
}
```

Reagieren auf eine Bildlaufaktion

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.ui.TouchEvent`
   - `net.rim.device.api.ui.TouchGesture`
   - `net.rim.device.api.ui.Field`
   - `net.rim.device.api.ui.Manager`
   - `net.rim.device.api.ui.Screen`
2. Erstellen Sie eine Klasse, die die `Manager`-Klasse erweitert.

```
 public class newManager extends Manager {
```

3. Prüfen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode, ob der von `TouchEvent.getEvent()` zurückgegebene Wert gleich `TouchEvent.MOVE` ist.

```
protected boolean touchEvent(TouchEvent message) {
   switch(message.getEvent()) {
      case TouchEvent.MOVE:
      return true;
   }
   return false;
}
```

Reagieren auf einen Benutzer, der den Bildschirm an zwei Stellen gleichzeitig berührt

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.ui.TouchEvent`
   - `net.rim.device.api.ui.Field`
   - `net.rim.device.api.ui.Manager`
   - `net.rim.device.api.ui.Screen`
   - `net.rim.device.api.ui.component.Dialog`
2. Erstellen Sie eine Klasse, die die `Manager`-Klasse, die `Screen`-Klasse, die `Field`-Klasse, oder eine der `Field`-Unterklassen erweitert.

   ```
   public class newButtonField extends ButtonField {
   ```

3. Prüfen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode, ob beim Aufrufen der folgenden Methoden Werte größer als Null zurückgegeben werden: `TouchEvent.getX(1)`, `TouchEvent.getY(1)`, `TouchEvent.getX(2)`, `TouchEvent.getY(2)`.

   ```
   protected boolean touchEvent(TouchEvent message) {
      switch(message.getEvent()) {
      case TouchEvent.MOVE:
         if(message.getX(1) > 0 && message.getY(1) > 0) {
            Dialog.alert("First finger touched/moved");
         }
         if(message.getX(2) > 0 && message.getY(2) > 0) {
            Dialog.alert("Second finger touched/moved");
         }
         return true;
      }
      return false;
   }
   ```

Ereignisinjektion

Mithilfe der `EventInjector`-Klasse und ihrer inneren Klassen können Sie programmgesteuert Benutzeroberflächenereignisse generieren. Auf BlackBerry®-Geräten, die BlackBerry® Device Software Version 5.0 oder höher ausführen und Touchscreens besitzen, können Sie Berührungsereignisse, wie z. B. Fingerstriche und Antippen, injizieren. Mithilfe einer der inneren `EventInjector`-Klassen können Sie ein Ereignis modellieren und mit der `invokeEvent()`-Methode können Sie das Ereignis injizieren. Das Ereignis wird an die Anwendung gesendet, die derzeit ausgewählt ist und auf die Eingabe wartet.

Sie können die Ereignisinjektion verwenden, um Tests zu automatisieren. Sie können die Ereignisinjektion auch verwenden, um Peripheriegeräten die Möglichkeit zu geben, mit einem BlackBerry-Gerät zu interagieren. Sie können sie auch in barrierefreien Anwendungen wie z. B. Sprache-in-Text-Konvertern verwenden. Um eine Beispielanwendung zu sehen, die die Ereignisinjektion vorführt, gehen Sie zu www.blackberry.com/go/toucheventinjectorsampleapp und laden Sie die Beispielanwendung des Berührungsereignisinjektors herunter. Weitere Informationen zur Beispielanwendung finden Sie unter www.blackberry.com/go/docs/developers in folgenden Dokumenten: *Touch Event Injector Sample Application Overview (Überblick über die Beispielanwendung des Berührungsereignisinjektors)*.

Anordnen von Benutzeroberflächenkomponenten 6

Sie können die Benutzeroberflächenkomponenten auf einem Anwendungsbildschirm mithilfe von BlackBerry®-API-Layout-Managern anordnen. Die folgenden Klassen erweitern die `Manager`-Klasse, die im `net.rim.device.apu.ui`-Paket bereitgestellt wird und stellen vordefinierte Layouts für die Benutzeroberflächenkomponenten auf dem Bildschirm Ihrer Anwendung bereit.

Layout-Manager	Beschreibung
`FlowFieldManager`	Dieser Layout-Manager ordnet Benutzeroberflächenkomponenten in Abhängigkeit von der Größe des Bildschirms vertikal und anschließend horizontal an. Die erste Benutzeroberflächenkomponente wird in der linken oberen Ecke des Bildschirms platziert und die folgenden Komponenten werden horizontal rechts von der ersten Komponente platziert, bis der Bildschirm über die gesamte Breite gefüllt ist. Wenn keine weiteren Benutzeroberflächenkomponenten in die erste Zeile passen, wird die nächste Benutzeroberflächenkomponente unter die erste Zeile von Komponenten in einer Zeile platziert, deren Höhe der Höhe der größten Komponente in der Zeile darüber entspricht. Sie können vertikale Stil-Bits (zum Beispiel `Field.FIELD_TOP`, `Field.FIELD_BOTTOM` oder `Field.FIELD_VCENTER`) anwenden, um Benutzeroberflächenkomponenten innerhalb ihrer Zeile vertikal auszurichten.
`HorizontalFieldManager`	Dieser Layout-Manager ordnet Benutzeroberflächenkomponenten in einer einzelnen horizontalen Zeile vom linken und zum rechten Rand des Bildschirms an. Weil dieser Layout-Manager Benutzeroberflächenkomponenten horizontal anordnet, können Sie auf die Benutzeroberflächenkomponenten keine horizontalen Stil-Bits (zum Beispiel `Field.FIELD_LEFT`, `Field.FIELD_HCENTER` oder `Field.FIELD_RIGHT`) anwenden. Sie können vertikale Stil-Bits (zum Beispiel `Field.FIELD_TOP`, `Field.FIELD_BOTTOM` oder `Field.FIELD_VCENTER`) anwenden. Wenn die Benutzeroberflächenkomponenten die verfügbare Breite des Bildschirm überschreiten, sollten Sie das `Manager.HORIZONTAL_SCROLL`-Stil-Bit verwenden, um ein horizontales Scrollen zu ermöglichen. Andernfalls werden auf dem Bildschirm so viele Benutzeroberflächenkomponenten wie innerhalb der

Layout-Manager	Beschreibung
	verfügbaren Bildschirmbreite möglich angezeigt und die übrigen werden nicht angezeigt. Die Benutzeroberflächenkomponenten sind vorhanden, aber nicht sichtbar. Dies kann zu einem unerwarteten Verhalten beim Scrollen führen.
`VerticalFieldManager`	Dieser Layout-Manager ordnet Benutzeroberflächenkomponenten auf dem Bildschirm von oben nach unten in einer einzelnen vertikalen Spalte an. Weil dieser Layout-Manager dafür entworfen wurde, Elemente vertikal anzuordnen, können Sie auf Benutzeroberflächenkomponenten keine vertikalen Stil-Bits (zum Beispiel `Field.FIELD_TOP`, `Field.FIELD_BOTTOM` oder `Field.FIELD_VCENTER`) anwenden. Sie können horizontale Stil-Bits (zum Beispiel `Field.FIELD_LEFT`, `Field.FIELD_HCENTER` oder `Field.FIELD_RIGHT`) anwenden.

Sie können zusätzliche Layout-Manager verwenden, um Benutzeroberflächenkomponenten in Ihrer Anwendung anzuordnen. Zum Beispiel können Sie den `GridFieldManager`-Layout-Manager verwenden, um Benutzeroberflächenkomponenten auf einem Bildschirm in Zeilen und Spalten zu platzieren und dadurch ein Raster zu erstellen. Sie können den `EyelidFieldManager`-Layout-Manager verwenden, um Benutzeroberflächenkomponenten in einem Paar von Managern anzuzeigen, die vorübergehend oben und unten auf dem Bildschirm angezeigt werden.

Anordnen von Benutzeroberflächenkomponenten

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
net.rim.device.api.ui.container.HorizontalFieldManager;
net.rim.device.api.ui.component.ButtonField;
```

2. Erstellen Sie eine Instanz von `HorizontalFieldManager`.

```
HorizontalFieldManager _fieldManagerBottom = new HorizontalFieldManager();
```

3. Rufen Sie `add()` auf, um dem Bildschirm den `HorizontalFieldManager` hinzuzufügen.

```
myScreen.add(_fieldManagerBottom);
```

4. Erstellen Sie eine Instanz von `ButtonField`.

```
ButtonField mySubmitButton = new ButtonField("Submit");
```

5. Fügen Sie dem `HorizontalFieldManager` den `ButtonField` hinzu.

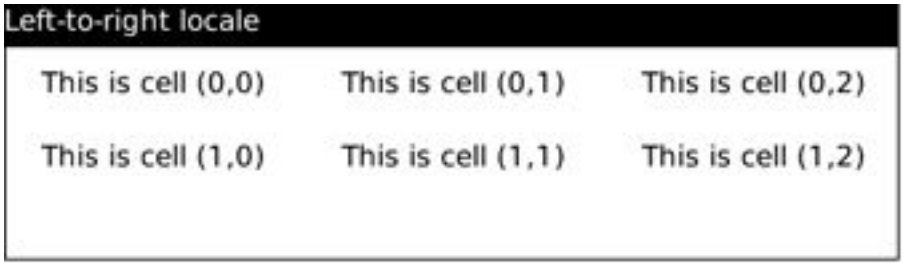
```
_fieldManagerBottom.add(mySubmitButton);
```

Erstellen eines Raster-Layouts

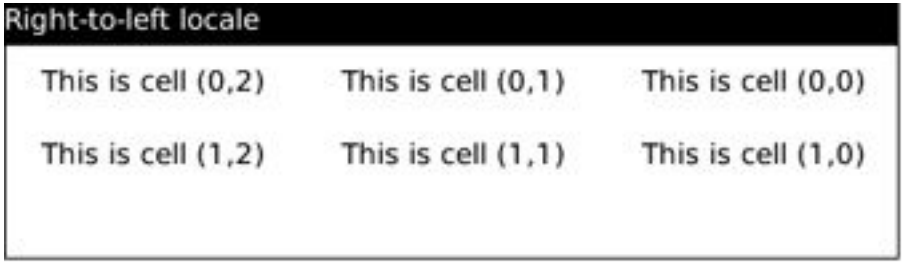
Anmerkung: Um Informationen zum Erstellen eines Raster-Layouts in der BlackBerry® Java® Development Environment vor Version 5.0 zu erhalten, gehen Sie zu http://www.blackberry.com/knowledgecenterpublic und lesen Sie DB-00783.

Sie können mithilfe der `GridFieldManager`-Klasse Felder auf einem Bildschirm in Zeilen und Spalten anordnen, um ein Raster zu erstellen. Wenn Sie ein Raster erstellen, können Sie die Anzahl der Zeilen und Spalten angeben. Nach dem Erstellen eines Rasters können Sie die Anzahl der Zeilen und Spalten, die es enthält, nicht mehr ändern.

Der Index von Rastern beginnt bei null. Deshalb befindet sich die erste Zelle bei Zeile 0, Spalte 0. In einem Gebietsschema mit einer Textrichtung von links nach rechts befindet sich die erste Zelle in der linken oberen Ecke des Rasters.



In einem Gebietsschema mit einer Textrichtung von rechts nach links befindet sich die erste Zelle in der rechten oberen Ecke des Rasters.



Sie können einem Raster Felder sequenziell hinzufügen (von links nach rechts oder von oben nach unten in Gebietsschemas mit einer Textrichtung von links nach rechts; von rechts nach links oder von oben nach unten in Gebietsschemas mit einer Textrichtung von rechts nach links) oder durch Angeben einer Zeile und einer Spalte im Raster. Sie können Felder löschen, Felder einfügen, die Abstände zwischen Spalten und Zeilen angeben und die Eigenschaften eines Rasters abrufen.

Raster haben keine definierten Überschriftenzeilen oder Überschriftenspalten. Sie können die Darstellung von Überschriften emulieren, indem Sie die Darstellung der Felder in der ersten Zeile oder der ersten Spalte des Rasters ändern.

Sie können horizontal und vertikal durch Raster scrollen, wenn die Breite oder Höhe des Rasters über den Bildschirmbereich hinausgeht.

Sie können durch Aufrufen von `GridFieldManager.setColumnProperty()` die Spaltenbreite und durch Aufrufen von `GridFieldManager.setRowProperty()` die Zeilenhöhe angeben. Wenn Sie diese Methoden aufrufen, müssen Sie eine `GridFieldManager`-Eigenschaft angeben.

Eigenschaft	Beschreibung
`FIXED_SIZE`	Breite oder Höhe ist ein fester Wert in Pixel
`PREFERRED_SIZE`	Breite oder Höhe ist eine bevorzugte Größe basierend auf der maximalen bevorzugten Größe der Felder in der Spalte oder der Zeile (`PREFERRED_SIZE` ist die Standardeigenschaft)
`PREFERRED_SIZE_WITH_MAXIMUM`	Breite oder Höhe ist eine bevorzugte Größe bis zu einer maximalen Größe
`AUTO_SIZE`	Breite oder Höhe basiert auf dem verfügbaren Platz auf dem Bildschirm

Erstellen einer Gittergröße

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Im Beispiel stellt die `GridScreen`-Klasse, beschrieben in Schritt 3, den benutzerdefinierten Bildschirm dar.

```
class GridFieldManagerDemo extends UiApplication
{
    public static void main(String[] args)
    {
        GridFieldManagerDemo theApp = new GridFieldManagerDemo();
        theApp.enterEventDispatcher();
    }
    GridFieldManagerDemo()
    {
        pushScreen(new GridScreen());
    }
}
```

3. Erstellen Sie das Framework für den benutzerdefinierten Bildschirm durch Erweitern der `MainScreen`-Klasse.

```
class GridScreen extends MainScreen
{
    public GridScreen()
```

```
    {
    }
}
```

4. Im Bildschirmkonstruktor rufen Sie `setTitle()` auf, um den Text festzulegen, den Sie im Titelabschnitt vom Bildschirm anzeigen wollen.

```
setTitle("GridFieldManager Demo");
```

5. Erstellen Sie im Bildschirmkonstruktor eine Instanz der `GridFieldManager`-Klasse. Geben Sie die Anzahl der Reihen, die Anzahl der Spalten und den Stil des Rasters an (mithilfe eines Stils, der von `net.rim.device.api.ui.Manager` geerbt wird). Geben Sie `0` für den Stil an, um den Standardstil zu verwenden.

```
GridFieldManager grid;
grid = new GridFieldManager(2,3,0);
```

6. Rufen Sie im Bildschirmkonstruktor `GridFieldManager.add()` auf, um Felder zum Raster hinzuzufügen.

```
grid.add(new LabelField("one"));
grid.add(new LabelField("two"));
grid.add(new LabelField("three"));
grid.add(new LabelField("four"));
grid.add(new LabelField("five"));
```

7. Rufen Sie im Bildschirmkonstruktor die `GridFieldManager Satz()`-Methoden auf, um die Eigenschaften vom Raster anzugeben.

```
grid.setColumnPadding(20);
grid.setRowPadding(20);
```

8. Rufen Sie im Bildschirmkonstruktor `Screen.add()` auf, um dem Bildschirm das Raster hinzuzufügen.

```
add(grid);
```

Nach Abschluss erforderliche Aktion:

Sie können das Raster ändern, nachdem Sie es erstellt haben. Zum Beispiel können Sie Felder hinzufügen, Felder löschen oder die Eigenschaften des Rasters ändern.

Codebeispiel: Erstellen eines Raster-Layouts

```
/*
 * GridFieldManagerDemo.java
 *
 * Research In Motion Limited proprietary and confidential
 * Copyright Research In Motion Limited, 2009
 */
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
public class GridFieldManagerDemo extends UiApplication
```

```
{
    public static void main(String[] args)
    {
        GridFieldManagerDemo theApp = new GridFieldManagerDemo();
        theApp.enterEventDispatcher();
    }
    GridFieldManagerDemo()
    {
        pushScreen(new GridScreen());
    }
}
class GridScreen extends MainScreen
{
    public GridScreen()
    {
        setTitle("GridFieldManager Demo");
        GridFieldManager grid = new GridFieldManager(2,3,0);
        grid.add(new LabelField("one"));
        grid.add(new LabelField("two"));
        grid.add(new LabelField("three"));
        grid.add(new LabelField("four"));
        grid.add(new LabelField("five"));
        grid.setColumnPadding(20);
        grid.setRowPadding(20);
        add(grid);
        // The grid looks like this:
        //   |  one   |  two     | three
        //   |  four  |  five    |
        // insert a cell first row, second column
        grid.insert(new LabelField("insert"), 0, 1);
        // The grid now looks like this:
        //   |  one   |  insert  | two
        //   |  three |  four    | five
        // delete a cell second row, second column
        grid.delete(1,1);
        // The grid now looks like this:
        //   |  one   |  insert  | two
        //   |  three |          | five
        // Add field to first unoccupied cell
        grid.add(new LabelField("new"));
        // The grid now looks like this:
        //   |  one   |  insert  | two
        //   |  three |   new    | five
    }
}
```

Anzeigen von Feldern in einem temporären Paar von Managern

Sie können die `EyelidFieldManager`-Klasse verwenden, um Felder in einem Paar von Managern anzuzeigen, die vorübergehend oben und unten auf dem Bildschirm angezeigt werden.

Standardmäßig werden die Felder angezeigt, wenn der BlackBerry®-Gerätebenutzer den Trackball bewegt oder, bei einem Gerät mit Touchscreen, den Bildschirm berührt. Die Felder werden nach einem bestimmten Zeitraum der Inaktivität (standardmäßig 1,2 Sekunden) ausgeblendet. Sie können diese Standardeigenschaften außer Kraft setzen.

Es gibt keine Beschränkung für die Anzahl und die Größe der Felder. Wenn die Manager mehr Felder enthalten, als auf den Bildschirm passen, überlappen der obere und der untere Manager einander, wobei sich der obere Manager im Vordergrund befindet.

Vorübergehendes Anzeigen eines ButtonField und eines LabelField oben bzw. unten am Bildschirm

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.extension.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `EyelidFieldManagerDemoScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class EyelidFieldManagerDemo extends UiApplication
{
     public static void main(String[] args)
     {
          EyelidFieldManagerDemo app = new EyelidFieldManagerDemo();
          app.enterEventDispatcher();
     }
     public EyelidFieldManagerDemo()
     {
          pushScreen(new EyelidFieldManagerDemoScreen());
     }
}
```

3. Erstellen Sie das Framework für den benutzerdefinierten Bildschirm durch Erweitern der `MainScreen`-Klasse.

```
class EyelidFieldManagerDemoScreen extends MainScreen {
{
     public EyelidFieldManagerDemoScreen()
     {
     }
}
```

4. Rufen Sie im Konstruktor `setTitle()` auf, um den Text festzulegen, der im Titelbereich des Bildschirms angezeigt wird.

```
setTitle("EyelidFieldManager Demo");
```

5. Erstellen Sie im Bildschirmkonstruktor eine Instanz der `EyelidFieldManager`-Klasse.

```
EyelidFieldManager manager = new EyelidFieldManager();
```

6. Rufen Sie im Bildschirmkonstruktor `EyelidFieldManager.addTop()` auf, um ein `LabelField`-Objekt dem oberen Manager des `EyelidFieldManager` hinzuzufügen.

```
manager.addTop(new LabelField("Hello World"));
```

7. Erstellen Sie im Bildschirmkonstruktor ein `HorizontalFieldManager`-Objekt. Rufen Sie `HorizontalFieldManager.add()` auf, um dem `HorizontalFieldManager` Schaltflächen hinzuzufügen. Rufen Sie `EyelidFieldManager.addBottom()` auf, um dem unteren Manager des `EyelidFieldManager` den `HorizontalFieldManager` hinzuzufügen.

```
HorizontalFieldManager buttonPanel = new HorizontalFieldManager
(Field.FIELD_HCENTER | Field.USE_ALL_WIDTH);
buttonPanel.add(new SimpleButton("Button 1"));
buttonPanel.add(new SimpleButton("Button 2"));
manager.addBottom(buttonPanel);
```

8. Rufen Sie im Bildschirmkonstruktor `EyelidFieldManager.setEyelidDisplayTime()` auf, um den Zeitraum der Inaktivität, der vergehen muss, bevor das Paar von Managern ausgeblendet wird in Sekunden anzugeben.

```
manager.setEyelidDisplayTime(3);
```

9. Rufen Sie im Bildschirmkonstruktor `add()` auf, um den `EyelidFieldManager` dem Bildschirm hinzuzufügen.

```
add(manager);
```

Codebeispiel: Vorübergehendes Anzeigen eines ButtonField und eines LabelField oben bzw. unten am Bildschirm

```
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.extension.container.*;
public class EyelidFieldManagerDemo extends UiApplication
{
     public static void main(String[] args)
```

```
     {
          EyelidFieldManagerDemo app = new EyelidFieldManagerDemo();
          app.enterEventDispatcher();
     }
     public EyelidFieldManagerDemo()
     {
          pushScreen(new EyelidFieldManagerDemoScreen());
     }
}
class EyelidFieldManagerDemoScreen extends MainScreen
{
     public EyelidFieldManagerDemoScreen()
     {
          setTitle(“EyelidFieldManager Demo”);
          EyelidFieldManager manager = new EyelidFieldManager();
          manager.addTop(new LabelField(“Hello World”));
          HorizontalFieldManager buttonPanel = new HorizonalFieldManager
(Field.FIELD_HCENTER | Field.USE_ALL_WIDTH);
          buttonPanel.add(new SimpleButton(“Button 1”));
          buttonPanel.add(new SimpleButton(“Button 2”));
          manager.addBottom(buttonPanel);
          manager.setEyelidDisplayTime(3);
          add(manager);
     }
}
```

Benutzeroberflächen-Komponenten 7

Hinzufügen einer Benutzeroberflächenkomponente zu einem Bildschirm

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.ui.component.CheckboxField`
   - `net.rim.device.api.ui.container.MainScreen`
2. Erstellen Sie eine Instanz einer Benutzeroberflächenkomponente.

```
CheckboxField myCheckbox = new CheckboxField("First checkbox", true);
```

3. Fügen Sie die Benutzeroberflächenkomponente Ihrer Erweiterung einer `Screen`-Klasse hinzu.

```
mainScreen.add(myCheckbox);
```

Ausrichten eines Feldes an einer Textzeile

Sie können mithilfe der Markierung `Field.FIELD_LEADING` eine Anwendung erstellen, die ein `Field`-Objekt am Anfang einer Textzeile ausrichten kann. Wenn Sie zum Beispiel ein `Field`-Objekt mit der Ausrichtungsmarkierung `Field.FIELD_LEADING` erstellen und einem `VerticalFieldManager`-Objekt das `Field`-Objekt hinzufügen, wird das `Field`-Objekt, wenn die Anwendung beispielsweise beginnt, ein englisches oder chinesisches Gebietsschemas zu verwenden, an der linken Seite des Bildschirms ausgerichtet. Wenn die Anwendung beginnt, ein arabisches oder hebräisches Gebietsschema zu verwenden, wird das `Field`-Objekt an der rechten Seite des Bildschirms ausgerichtet.

Autocomplete-Textfeld

Sie können ein autocomplete-Textfeld verwenden, um vorherzusagen, was ein BlackBerry®-Gerätebenutzer eingeben möchte, und ein Wort bzw. einen Satz anzuzeigen, bevor der Benutzer dessen Eingabe abgeschlossen hat.

Wenn Sie ein `AutoCompleteField`-Objekt erstellen, müssen Sie ein `BasicFilteredList`-Objekt damit verknüpfen. Durch `BasicFilteredList` bleiben die Bezüge auf Datenobjekte erhalten, mit denen verglichen wird, um die Liste der Wörter und Sätze zu erstellen. Sie können in der Konfiguration festlegen, mit welchen Feldern in den Datenobjekten verglichen wird, und welche Felder angezeigt werden, wenn ein Gegenstück gefunden wird. So können Sie beispielsweise den Text, den der Benutzer eingibt mit dem Wert des Felds `DATA_FIELD_CONTACTS_BIRTHDAY` in der Datenquelle `DATA_SOURCE_CONTACTS` vergleichen, und den Wert des entsprechenden Felds `DATA_FIELD_CONTACTS_NAME_FULL` zurückgeben.

Es gibt vier Datentypen, die Sie an eine `BasicFilteredList` binden können, um diese mit einem `AutoCompleteField` zu verwenden.

Sie können den Satz der Zeichenfolgen angeben, mit dem auf eine der folgenden Arten verglichen wird:

- Ein Array wörtlicher Zeichenfolgen
- Ein Array von Objekten, das `toString()` unterstützt
- Datenquellen auf einem BlackBerry-Gerät, wie z. B. Kontakte, Notizen, Aufgaben und verschiedene Arten von Mediendateien
- Ein Array von Objekten und einem Array von Zeichenfolgen mit entsprechenden Indizes

Standardmäßig zeigt das autocomplete-Textfeld den Satz der Zeichenfolgen an, der vom Vergleichsprozess in einer Dropdown-Liste zurückgegeben wird. Sie können das Erscheinungsbild dieser Liste durch Angeben von Stilmarkierungen bei der Erstellung des autocomplete-Textfeldes konfigurieren. Sie können ändern, wie die Liste anzeigt wird, und wie Benutzer mit der Liste interagieren können.

Erstellen eines autocomplete-Textfelds aus einem Datensatz

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.UiApplication;
import net.rim.device.api.ui.container.MainScreen;
import net.rim.device.api.ui.component.AutoCompleteField;
import net.rim.device.api.collection.util.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `HomeScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class AutoCompleteFieldApp extends UiApplication
{
    public static void main(String[] args)
    {
        AutoCompleteFieldApp app = new AutoCompleteFieldApp();
        app.enterEventDispatcher();
    }
    AutoCompleteFieldApp()
    {
            pushScreen(new HomeScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm durch Erweitern der `MainScreen`-Klasse.

```
class HomeScreen extends MainScreen
{
    public HomeScreen()
    {
    }
}
```

4. Erstellen Sie im Konstruktor ein `BasicFilteredList`-Objekt. Erstellen Sie ein `String`-Array und speichern Sie die Zeichenfolgen, die sich im Array entsprechen sollen. In diesem Beispiel sind die Zeichenfolgen Wochentage. Rufen Sie `addDataSet()` auf, um die Daten im Array an `BasicFilteredList` zu binden.

```
BasicFilteredList filterList = new BasicFilteredList();
String[] days =
{"Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunda
    "};
filterList.addDataSet(1,days,"days",BasicFilteredList.COMPARISON_IGNORE_CASE);
```

5. Erstellen Sie im Konstruktor ein `AutoCompleteField`-Objekt. Übergeben Sie eine Instanz der `BasicFilteredList` an den `AutoCompleteField`-Konstruktor, um die `BasicFilteredList` an das Autocomplete-Textfeld zu binden. Rufen Sie `add()` auf, um das Feld dem Bildschirm hinzuzufügen.

```
AutoCompleteField autoCompleteField = new
    AutoCompleteField(filterList);
add(autoCompleteField);
```

Codebeispiel: Erstellen eines autocomplete-Felds aus einem Datensatz

```
import net.rim.device.api.ui.UiApplication;
import net.rim.device.api.ui.container.MainScreen;
import net.rim.device.api.ui.component.AutoCompleteField;
import net.rim.device.api.collection.util.*;
public class AutoCompleteFieldApp extends UiApplication
{
    public static void main(String[] args)
    {
        AutoCompleteFieldApp app = new AutoCompleteFieldApp();
        app.enterEventDispatcher();
    }
    AutoCompleteFieldApp()
    {
           HomeScreen scr = new HomeScreen();
           this.pushScreen(scr);
    }
}
class HomeScreen extends MainScreen
{
    public HomeScreen()
    {
        BasicFilteredList filterList = new BasicFilteredList();
        String[] days = {"Monday","Tuesday","Wednesday",
                         "Thursday","Friday","Saturday","Sunday"};
        filterList.addDataSet(1,days,"days",BasicFilteredList
    COMPARISON_IGNORE_CASE);
        AutoCompleteField autoCompleteField = new AutoCompleteField(filterList);
```

```
        add(autoCompleteField);
    }
}
```

Erstellen eines autocomplete-Textfelds aus einer Datenquelle

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.UiApplication;
import net.rim.device.api.ui.container.MainScreen;
import net.rim.device.api.ui.component.AutoCompleteField;
import net.rim.device.api.collection.util.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die `HomeScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar, der in Schritt 3 beschrieben wird.

```
public class AutoCompleteFieldApp extends UiApplication
{
    public static void main(String[] args)
    {
        AutoCompleteFieldApp app = new AutoCompleteFieldApp();
        app.enterEventDispatcher();
    }
    public AutoCompleteFieldApp()
    {
            pushScreen(new HomeScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse.

```
class HomeScreen extends MainScreen
{
    public HomeScreen()
    {
    }
}
```

4. Erstellen Sie im Bildschirmkonstruktor ein `BasicFilteredList`-Objekt. Rufen Sie `addDataSource()` auf, um eine Datenquelle an `BasicFilteredList` zu binden. In diesem Beispiel handelt es sich bei den Daten um Kontaktinformationen und die Datenquelle ist die Kontaktliste. Das erste Argument, das Sie an `addDataSource()` übergeben, ist eine eindeutige ID. Das zweite Argument bindet das `BasicFilteredList`-Objekt an eine Datenquelle. Das dritte Argument gibt den Satz von Datenquellenfeldern an, mit denen verglichen wird. Das vierte Argument gibt den Satz von Datenquellenfeldern an, die bereitgestellt werden sollen, wenn eine Übereinstimmung gefunden wird. In diesem Beispiel sind die Felder, mit denen verglichen wird, identisch mit den Feldern, die bereitgestellt werden sollen, wenn eine

Übereinstimmung gefunden wird. Das fünfte Argument gibt das primäre Anzeigefeld an. Das sechste Argument gibt das sekundäre Anzeigefeld an und ist auf -1 gesetzt, wenn Sie es nicht verwenden möchten. Das letzte Argument gibt einen Namen für `BasicFilteredList` an. Sein Wert wird automatisch generiert, wenn Sie `null` angeben.

```
BasicFilteredList filterList = new BasicFilteredList();
        filterList.addDataSource(
            1,
            BasicFilteredList.DATA_SOURCE_CONTACTS,
            BasicFilteredList.DATA_FIELD_CONTACTS_NAME_FULL |
            BasicFilteredList.DATA_FIELD_CONTACTS_COMPANY |
            BasicFilteredList.DATA_FIELD_CONTACTS_EMAIL,
            BasicFilteredList.DATA_FIELD_CONTACTS_NAME_FULL |
            BasicFilteredList.DATA_FIELD_CONTACTS_COMPANY |
            BasicFilteredList.DATA_FIELD_CONTACTS_EMAIL,
            BasicFilteredList.DATA_FIELD_CONTACTS_NAME_FULL,
            -1,
            null);
```

5. Erstellen Sie im Bildschirmkonstruktor ein `AutoCompleteField`-Objekt. Übergeben Sie das `BasicFilteredList`-Objekt, das Sie in Schritt 4 erstellt haben, an den `AutoCompleteField`-Konstruktor, um `BasicFilteredList` an das autocomplete-Textfeld zu binden. Rufen Sie `add()` auf, um das Feld dem Bildschirm hinzuzufügen.

```
AutoCompleteField autoCompleteField = new
    AutoCompleteField(filterList);
add(autoCompleteField);
```

Codebeispiel: Erstellen eines autocomplete-Felds aus einer Datenquelle

```
import net.rim.device.api.ui.UiApplication;
import net.rim.device.api.ui.container.MainScreen;
import net.rim.device.api.ui.component.AutoCompleteField;
import net.rim.device.api.collection.util.*;
public class AutoCompleteFieldApp extends UiApplication
{
    public static void main(String[] args)
    {
         AutoCompleteFieldApp app = new AutoCompleteFieldApp();
         app.enterEventDispatcher();
    }
       AutoCompleteFieldApp()
       {
           HomeScreen scr = new HomeScreen();
           this.pushScreen(scr);
       }
}
class HomeScreen extends MainScreen
{
    public HomeScreen()
    {
        BasicFilteredList filterList = new BasicFilteredList();
```

```
        filterList.addDataSource(1,
            BasicFilteredList.DATA_SOURCE_CONTACTS,
            BasicFilteredList.DATA_FIELD_CONTACTS_NAME_FULL |
BasicFilteredList.DATA_FIELD_CONTACTS_COMPANY |
BasicFilteredList.DATA_FIELD_CONTACTS_EMAIL,
            BasicFilteredList.DATA_FIELD_CONTACTS_NAME_FULL |
BasicFilteredList.DATA_FIELD_CONTACTS_COMPANY |
BasicFilteredList.DATA_FIELD_CONTACTS_EMAIL,
            BasicFilteredList.DATA_FIELD_CONTACTS_NAME_FULL,
            BasicFilteredList.DATA_FIELD_CONTACTS_NAME_FULL,
            null);
        AutoCompleteField autoCompleteField = new AutoCompleteField(filterList);
        add(autoCompleteField);
    }
}
```

Verwenden von Datenquellen und Feldern mit einem autocomplete-Textfeld

Mithilfe der Klassen `AutoCompleteField` und `BasicFilteredList` können Sie den Text, den ein Benutzer in einem autocomplete-Textfeld eingibt, mit den Werten von Feldern in einer angegebenen Datenquelle vergleichen. Sie geben die zu verwendenden Felder und ihre Datenquellen mithilfe eines `BasicFilteredList`-Objekts an, das Sie als Argument an den Konstruktor der `AutoCompleteField`-Klasse übergeben.

Datenquelle	Felder
DATA_SOURCE_APPOINTMENTS	• DATA_FIELD_APPOINTMENTS_ALL • DATA_FIELD_APPOINTMENTS_ATTENDEES • DATA_FIELD_APPOINTMENTS_ORGANIZER • DATA_FIELD_APPOINTMENTS_SUBJECT
DATA_SOURCE_CONTACTS	• DATA_FIELD_CONTACTS_ADDRESS_ALL • DATA_FIELD_CONTACTS_ADDRESS_HOME • DATA_FIELD_CONTACTS_ADDRESS_WORK • DATA_FIELD_CONTACTS_ANNIVERSARY • DATA_FIELD_CONTACTS_BIRTHDAY • DATA_FIELD_CONTACTS_CATEGORIES • DATA_FIELD_CONTACTS_COMPANY • DATA_FIELD_CONTACTS_EMAIL • DATA_FIELD_CONTACTS_FAX • DATA_FIELD_CONTACTS_JOB_TITLE • DATA_FIELD_CONTACTS_NAME_FULL

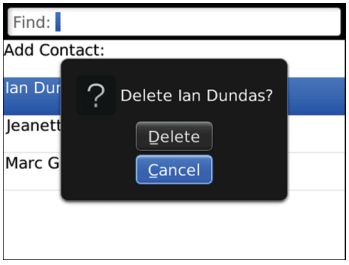
Datenquelle	Felder
	• DATA_FIELD_CONTACTS_NAME_FIRST • DATA_FIELD_CONTACTS_NAME_LAST • DATA_FIELD_CONTACTS_NOTES • DATA_FIELD_CONTACTS_PAGER • DATA_FIELD_CONTACTS_PHONE_ALL • DATA_FIELD_CONTACTS_PHONE_HOME • DATA_FIELD_CONTACTS_PHONE_HOME2 • DATA_FIELD_CONTACTS_PHONE_MOBILE • DATA_FIELD_CONTACTS_PHONE_OTHER • DATA_FIELD_CONTACTS_PHONE_WORK • DATA_FIELD_CONTACTS_PHONE_WORK2 • DATA_FIELD_CONTACTS_PIN
DATA_SOURCE_MEMOS	• DATA_FIELD_MEMOS_TITLE
DATA_SOURCE_MESSAGES	• DATA_FIELD_MESSAGES_ALL • DATA_FIELD_MESSAGES_RECIPIENT • DATA_FIELD_MESSAGES_SENDER • DATA_FIELD_MESSAGES_SUBJECT
DATA_SOURCE_MUSIC	• DATA_FIELD_MUSIC_ALL • DATA_FIELD_MUSIC_ALBUM • DATA_FIELD_MUSIC_ARTIST • DATA_FIELD_MUSIC_GENRE • DATA_FIELD_MUSIC_PLAYLIST • DATA_FIELD_MUSIC_SONG
DATA_SOURCE_PICTURES	• DATA_FIELD_PICTURES_TITLE
DATA_SOURCE_RINGTONES	• DATA_FIELD_RINGTONES_TITLE
DATA_SOURCE_TASKS	• DATA_FIELD_TASKS_TITLE
DATA_SOURCE_VIDEOS	• DATA_FIELD_VIDEOS_TITLE
DATA_SOURCE_VOICENOTES	• DATA_FIELD_VOICENOTES_TITLE

Schaltflächen

Verwenden Sie Schaltflächen, um Benutzern zu ermöglichen, eine Aktion aus einem Dialogfeld auszuführen. Menüs enthalten in der Regel Aktionen, die mit einem Bildschirm zusammenhängen.

Benutzer können über eine Schaltfläche die folgenden Aktionen ausführen:

Aktion	BlackBerry-Geräte ohne Touchscreen	BlackBerry-Geräte mit Touchscreen
Markieren Sie eine Schaltfläche.	• Drehen Sie das Trackwheel oder den Trackball. • Fahren Sie mit einem Finger über das Trackpad.	Berühren Sie die Schaltfläche leicht.
Führen Sie eine Aktion aus.	Klicken Sie auf die Schaltfläche oder drücken Sie die Eingabetaste.	Klicken Sie auf den Bildschirm.



Bewährtes Verfahren: Implementieren von Schaltflächen

- Vermeiden Sie auf einem Anwendungsbildschirm den Einsatz von Schaltflächen. Zur Bereitstellung von mit einem Bildschirm verknüpften Aktionen, verwenden Sie stattdessen wo möglich das Anwendungsmenü. Das Menü steht Benutzern sofort zur Verfügung, unabhängig von der Position des Cursors auf dem Bildschirm. Schaltflächen sind statisch und erfordern vom Benutzer eine Markierung der Schaltfläche, um die verknüpfte Aktion auszuführen. Wenn Sie Schaltflächen verwenden, nehmen Sie ebenfalls Menüelemente für die Aktionen im Anwendungsmenü auf.
- Verwenden Sie wo möglich Kontrollkästchen oder Optionsfelder für Optionen wie z. B. das Ein- oder Ausschalten einer Funktion.
- Verwenden Sie für die Standardschaltfläche jene Schaltfläche, die Benutzer am wahrscheinlichsten auswählen. Vermeiden Sie die Verwendung einer Schaltfläche, die mit einer beendenden Aktion verknüpft ist, als Standardschaltfläche.

Richtlinien für Bezeichnungen

- Verwenden Sie kurze, klare Bezeichnungen.

- Verwenden Sie nach Möglichkeit Bezeichnungen, die aus einem Wort bestehen. Die Größe einer Schaltfläche ändert sich mit der Länge der Bezeichnung. Wenn eine Bezeichnung zu lang ist, werden Auslassungspunkte (...) angezeigt, um zu verdeutlichen, dass der Text abgeschnitten ist. Wenn Sie eine Bezeichnung aus mehreren Wörtern verwenden, verwenden Sie satzabhängige Großschreibung.
- Verwenden Sie für Bezeichnungen Verben, die die verknüpfte Aktion beschreiben (zum Beispiel "Abbrechen", "Löschen", "Verwerfen" oder "Speichern"). Falls erforderlich, fügen Sie weitere Beschreibungen anderswo auf dem Bildschirm hinzu (zum Beispiel in einer Anwendungsnachricht).
- Vermeiden Sie die Verwendung der Bezeichnungen "Ja" und "Nr."
- Vermeiden Sie die Verwendung von Symbolen oder Grafik in einer Bezeichnung.
- Vermeiden Sie Interpunktion in einer Bezeichnung. Verwenden Sie in der Bezeichnung einer Schaltfläche Auslassungspunkte um anzuzeigen, dass Benutzer eine weitere Aktion ausführen müssen, nachdem sie auf die Schaltfläche klicken.

Erstellen einer Schaltfläche

1. Importieren Sie die `net.rim.device.api.ui.component.ButtonField`-Klasse.
2. Erstellen Sie mithilfe eines Stilparameters eine Instanz eines `ButtonField`-Objekts.

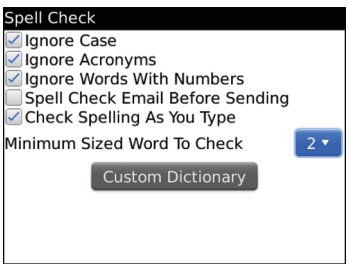
```
ButtonField mySubmitButton = new ButtonField("Submit");
```

Kontrollkästchen

Verwenden Sie Kontrollkästchen für Optionen, die Benutzer aktivieren oder deaktivieren können.

Benutzer können mit einem Kontrollkästchen die folgende Aktion ausführen:

Aktion	BlackBerry-Geräte ohne Touchscreen	BlackBerry-Geräte mit Touchscreen
Wählen Sie ein Kontrollkästchen aus.	Drücken Sie die Leertaste oder klicken Sie auf das Trackwheel, den Trackball oder das Trackpad.	Klicken Sie auf den Bildschirm.



Bewährtes Verfahren: Implementieren von Kontrollkästchen

- Verwenden Sie Kontrollkästchen, wenn Benutzer mehrere Optionen auswählen können.
- Verwenden Sie die `CheckboxField`-Klasse, um Kontrollkästchen zu erstellen.
- Starten Sie keine Aktion, wenn Benutzer ein Kontrollkästchen markieren. Zum Beispiel keinen neuen Bildschirm öffnen.
- Richten Sie die Kontrollkästchen senkrecht aus.
- Gruppieren und ordnen Sie Kontrollkästchen logisch an (gruppieren Sie beispielsweise zusammen gehörende Objekte oder führen Sie die häufigsten Objekte zuerst an). Vermeiden Sie eine alphabetische Sortierung der Kontrollkästchen; die alphabetische Reihenfolge ist sprachspezifisch.

Richtlinien für Bezeichnungen

- Verwenden Sie kurze, klare Bezeichnungen. Überprüfen Sie, ob die Bezeichnung deutlich beschreibt, was geschieht, wenn Benutzer das Kontrollkästchen aktivieren.
- Verwenden Sie nach Möglichkeit positive Bezeichnungen. Wenn beispielsweise Benutzer die Option zum Ein-oder Ausschalten einer Funktion haben, verwenden Sie in der Bezeichnung "Einschalten" statt "Ausschalten".
- Platzieren Sie die Bezeichnungen auf der rechten Seite von Kontrollkästchen.
- Verwenden Sie die übliche Groß-/Kleinschreibung für Titel.
- Verwenden Sie keine Ende-Interpunktion.

Erstellen eines Kontrollkästchens

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `MyUiScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class MyUi extends UiApplication
{
    public static void main(String[] args)
    {
        MyUi theApp = new MyUi();
        theApp.enterEventDispatcher();
    }
    public MyUi()
    {
        pushScreen(new MyUiScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Bildschirmkonstruktor `setTitle()` auf, um den Titel für den Bildschirm anzugeben.

```
class MyUiScreen extends MainScreen
{
    public MyUiScreen()
    {
        setTitle("UI Component Sample");
    }
}
```

4. Erstellen Sie im Bildschirmkonstruktor ein Kontrollkästchen mithilfe der `CheckboxField`-Klasse. Geben Sie im `CheckboxField`-Konstruktor die Bezeichnung für das Kontrollkästchen an und verwenden Sie einen booleschen Wert, um anzuzeigen, ob das Kontrollkästchen die Standardauswahl ist (`true` zeigt zum Beispiel an, dass das Kontrollkästchen standardmäßig ausgewählt ist). Rufen Sie `add()` auf, um dem Bildschirm das Kontrollkästchen hinzuzufügen.

```
add(new CheckboxField("First Check Box", true));
add(new CheckboxField("Second Check Box", false));
```

5. Um das Standardverhalten eines Kontrollkästchens zu ändern, verwenden Sie die Stile, die von der `Field`-Klasse geerbt wurden. Um zum Beispiel ein Kontrollkästchen zu erstellen, dass die Benutzer nicht ändern können, verwenden Sie den `READONLY`-Stil.

```
add(new CheckboxField("First Check Box", true, this.READONLY));
```

6. Um die Standardfunktionalität außer Kraft zu setzen, die den Benutzer auffordert, Änderungen vor dem Schließen der Anwendung zu speichern, setzen Sie in der Erweiterung der `MainScreen`-Klasse die `MainScreen.onSavePrompt()`-Methode außer Kraft. Im folgenden Codebeispiel lautet der Rückgabewert `true`, wodurch angezeigt wird, dass die Anwendung den Benutzer vor dem Schließen nicht auffordert.

```
public boolean onSavePrompt()
{
    return true;
}
```

Codebeispiel: Erstellen eines Kontrollkästchens

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
public class MyUi extends UiApplication
{
    public static void main(String[] args)
    {
        MyUi theApp = new MyUi();
        theApp.enterEventDispatcher();
    }
    public MyUi()
    {
```

```
        pushScreen(new MyUiScreen());
    }
}
class MyUiScreen extends MainScreen
{
    public MyUiScreen()
    {
        setTitle("UI Component Sample");
        add(new CheckboxField("First Check Box", true));
        add(new CheckboxField("Second Check Box", false));
    }
    public boolean onSavePrompt()
    {
        return true;
    }
}
```

Erstellen einer Bitmap

1. Importieren Sie die `net.rim.device.api.ui.component.BitmapField`-Klasse.
2. Erstellen Sie eine Instanz von `BitmapField`.

```
BitmapField myBitmapField = new BitmapField();
```

Erstellen eines benutzerdefinierten Feldes

Sie können einem benutzerdefinierten Feld nur benutzerdefinierte Kontextmenüeinträge und benutzerdefinierte Layouts hinzufügen.

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.ui.Field`
   - `java.lang.String`
   - `net.rim.device.api.ui.Font`
   - `java.lang.Math`
   - `net.rim.device.api.ui.Graphics`
2. Importieren Sie die `net.rim.device.api.ui.DrawStyle`-Schnittstelle.
3. Erweitern Sie die `Field`-Klasse oder eine ihrer Unterklassen, die die `DrawStyle`-Schnittstelle implementieren, um die Eigenschaften des benutzerdefinierten Feldes anzugeben und die Zeichnungsstile zu aktivieren.

```
public class CustomButtonField extends Field implements DrawStyle {
   public static final int RECTANGLE = 1;
   public static final int TRIANGLE = 2;
   public static final int OCTAGON = 3;
   private String _label;
```

```
    private int _shape;
    private Font _font;
    private int _labelHeight;
    private int _labelWidth;
}
```

4. Implementieren Sie Konstruktoren, um die Bezeichnung, die Form und den Stil der benutzerdefinierten Schaltfläche zu definieren.

```
public CustomButtonField(String label) {
    this(label, RECTANGLE, 0);
}
public CustomButtonField(String label, int shape) {
    this(label, shape, 0);
}
public CustomButtonField(String label, long style) {
    this(label, RECTANGLE, style);
}
public CustomButtonField(String label, int shape, long style) {
    super(style);
    _label = label;
    _shape = shape;
    _font = getFont();
    _labelHeight = _font.getHeight();
    _labelWidth = _font.getAdvance(_label);
}
```

5. Implementieren Sie `layout()`, um die Anordnung der Felddaten anzugeben. Führen Sie die komplexesten Berechnungen in `layout()` statt in `paint()` aus. Der Manager des Feldes ruft layout() auf, um zu bestimmen, wie das Feld seine Inhalte im verfügbaren Platz anordnet. Rufen Sie `Math.min()` auf, um die kleineren Werte für die angegebene Breite und Höhe sowie die bevorzugte Breite und Höhe des Feldes zurückzugeben. Rufen Sie `Field.setExtent(int,int)` auf, um die erforderlichen Abmessungen für das Feld festzulegen.

```
protected void layout(int width, int height) {
    _font = getFont();
    _labelHeight = _font.getHeight();
    _labelWidth = _font.getAdvance(_label);
    width = Math.min( width, getPreferredWidth() );
    height = Math.min( height, getPreferredHeight() );
    setExtent( width, height );
}
```

6. Implementieren Sie `getPreferredWidth()` mithilfe der relativen Maßangaben der Feldbezeichnung, um sicherzustellen, dass die Bezeichnung die Abmessungen der Komponente nicht überschreitet. Verwenden Sie einen switch-Block, um die bevorzugte Breite basierend auf der Form des benutzerdefinierten Feldes zu bestimmen. Verwenden Sie für jeden Formentyp eine if-Anweisung, um Maßangaben zu vergleichen und die bevorzugte Breite für das benutzerdefinierte Feld zu bestimmen.

```
public int getPreferredWidth() {
    switch(_shape) {
       case TRIANGLE:
```

```
            if (_labelWidth < _labelHeight) {
               return _labelHeight << 2;
            } else {
               return _labelWidth << 1;
            }
         case OCTAGON:
            if (_labelWidth < _labelHeight) {
               return _labelHeight + 4;
            } else {
               return _labelWidth + 8;
            }
         case RECTANGLE: default:
            return _labelWidth + 8;
      }
   }
```

7. Implementieren Sie `getPreferredHeight()` mithilfe der relativen Maßangaben der Feldbezeichnung, um die bevorzugte Höhe zu bestimmen. Verwenden Sie einen switch-Block, um die bevorzugte Höhe basierend auf der Form des benutzerdefinierten Feldes zu bestimmen. Verwenden Sie für jeden Formentyp eine if-Anweisung, um Maßangaben zu vergleichen und die bevorzugte Höhe für das benutzerdefinierte Feld zu bestimmen.

```
public int getPreferredHeight() {
   switch(_shape) {
      case TRIANGLE:
       if (_labelWidth < _labelHeight) {
          return _labelHeight << 1;
       } else {
          return _labelWidth;
       }
      case RECTANGLE:
       return _labelHeight + 4;
       case OCTAGON:
       return getPreferredWidth();
   }
   return 0;
}
```

8. Implementieren Sie `paint()`. Der Manager eines Feldes ruft `paint()` auf, um das Feld neu zu zeichnen, wenn ein Bereich des Feldes als ungültig markiert wird. Verwenden Sie einen switch-Block, um das Feld basierend auf der Form des benutzerdefinierten Feldes neu zu zeichnen. Verwenden bei einem Feld mit einer dreieckigen oder achteckigen Form die Breite des Feldes, um die horizontale und vertikale Position des Anfangs- und Endpunkts einer Linie zu berechnen. Rufen Sie `graphics.drawLine()` auf und verwenden Sie die Anfangs- und Endpunkte, um die Linien zu zeichnen, die das benutzerdefinierte Feld definieren. Rufen Sie bei einem Feld mit einer rechteckigen Form `graphics.drawRect()` auf und verwenden Sie die Breite und die Höhe des Feldes, um das benutzerdefinierte Feld zu zeichnen. Rufen Sie `graphics.drawText()` auf und verwenden Sie die Breite des Feldes, um eine Textzeichenfolge in einen Bereich des Feldes zu zeichnen.

```
protected void paint(Graphics graphics) {
   int textX, textY, textWidth;
   int w = getWidth();
```

```
   switch(_shape) {
      case TRIANGLE:
       int h = (w>>1);
       int m = (w>>1)-1;
       graphics.drawLine(0, h-1, m, 0);
       graphics.drawLine(m, 0, w-1, h-1);
       graphics.drawLine(0, h-1, w-1, h-1);
       textWidth = Math.min(_labelWidth,h);
       textX = (w - textWidth) >> 1;
       textY = h >> 1;
       break;
      case OCTAGON:
       int x = 5*w/17;
       int x2 = w-x-1;
       int x3 = w-1;
       graphics.drawLine(0, x, 0, x2);
       graphics.drawLine(x3, x, x3, x2);
       graphics.drawLine(x, 0, x2, 0);
       graphics.drawLine(x, x3, x2, x3);
       graphics.drawLine(0, x, x, 0);
       graphics.drawLine(0, x2, x, x3);
       graphics.drawLine(x2, 0, x3, x);
       graphics.drawLine(x2, x3, x3, x2);
       textWidth = Math.min(_labelWidth, w - 6);
       textX = (w-textWidth) >> 1;
       textY = (w-_labelHeight) >> 1;
       break;
      case RECTANGLE: default:
       graphics.drawRect(0, 0, w, getHeight());
       textX = 4;
       textY = 2;
       textWidth = w - 6;
       break;
   }
   graphics.drawText(_label, textX, textY, (int)( getStyle() & DrawStyle.ELLIPSIS
| DrawStyle.HALIGN_MASK ), textWidth );
}
```

9. Implementieren Sie die Methoden `Field set()` und `get()`. Implementieren Sie die Methoden `Field.getLabel()`, `Field.getShape()`, `Field.setLabel(String label)` und `Field.setShape(int shape)`, um die Instanzvariablen des benutzerdefinierten Feldes zurückzugeben.

```
public String getLabel() {
   return _label;
}
public int getShape() {
   return _shape;
}
public void setLabel(String label) {
   _label = label;
   _labelWidth = _font.getAdvance(_label);
   updateLayout();
```

```
}
public void setShape(int shape) {
   _shape = shape;
}
```

Erstellen eines Felds, um Webinhalte anzuzeigen

Anzeigen von HTML-Inhalt in einem Browserfeld

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.browser.field2.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `BrowserFieldDemoScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class BrowserFieldDemo extends UiApplication
{
     public static void main(String[] args)
     {
          BrowserFieldDemo app = new BrowserFieldDemo();
          app.enterEventDispatcher();
     }
     public BrowserFieldDemo()
     {
          pushScreen(new BrowserFieldDemoScreen());
     }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm durch Erweitern der `MainScreen`-Klasse.

```
class BrowserFieldDemoScreen extends MainScreen
{
     public BrowserFieldDemoScreen()
     {
     }
}
```

4. Erstellen Sie im Bildschirmkonstruktor eine Instanz der `BrowserField`-Klasse.

```
BrowserField myBrowserField = new BrowserField();
```

5. Rufen Sie im Bildschirmkonstruktor `add()` auf, um dem Bildschirm das `BrowserField`-Objekt hinzuzufügen.

```
add(myBrowserField);
```

6. Rufen Sie im Bildschirmkonstruktor `BrowserField.displayContent()` auf, um den HTML-Inhalt anzugeben und anzuzeigen.

```
myBrowserField.displayContent("<html><body><h1>Hello
    World!</h1></body></html>", "http://localhost");
```

Codebeispiel: Anzeigen von HTML-Inhalt in einem Browserfeld

```
import net.rim.device.api.browser.field2.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
public class BrowserFieldDemo extends UiApplication
{
    public static void main(String[] args)
    {
        BrowserFieldDemo app = new BrowserFieldDemo();
        app.enterEventDispatcher();
    }
    public BrowserFieldDemo()
    {
        pushScreen(new BrowserFieldDemoScreen());
    }
}
class BrowserFieldDemoScreen extends MainScreen
{
    public BrowserFieldDemoScreen()
    {
        BrowserField myBrowserField = new BrowserField();
        add(myBrowserField);
        myBrowserField.displayContent("<html><body><h1>Hello
    World!</h1></body></html>", "http://localhost");
    }
}
```

Anzeigen von HTML-Inhalt von einer Webseite in einem Browserfeld

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.browser.field2.*;
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `BrowserFieldDemoScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class BrowserFieldDemo extends UiApplication
{
     public static void main(String[] args)
     {
          BrowserFieldDemo app = new BrowserFieldDemo();
          app.enterEventDispatcher();
     }
     public BrowserFieldDemo()
     {
          pushScreen(new BrowserFieldDemoScreen());
     }
}
```

3. Erstellen Sie das Framework für den benutzerdefinierten Bildschirm durch Erweitern der `MainScreen`-Klasse.

```
class BrowserFieldDemoScreen extends MainScreen
{
     public BrowserFieldDemoScreen()
     {
     }
}
```

4. Erstellen Sie im Bildschirmkonstruktor eine Instanz der `BrowserField`-Klasse.

```
BrowserField myBrowserField = new BrowserField();
```

5. Rufen Sie im Bildschirmkonstruktor `add()` auf, um dem Bildschirm das `BrowserField`-Objekt hinzuzufügen.

```
add(myBrowserField);
```

6. Rufen Sie im Bildschirmkonstruktor `BrowserField.requestContent()` auf, um den Ort des HTML-Inhalts anzugeben und ihn anzuzeigen.

```
myBrowserField.requestContent("http://www.blackberry.com");
```

Codebeispiel: Anzeigen von HTML-Inhalt von einer Webseite in einem Browserfeld

```
import net.rim.device.api.browser.field2.*;
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
public class BrowserFieldDemo extends UiApplication
{
    public static void main(String[] args)
    {
```

```
        BrowserFieldDemo app = new BrowserFieldDemo();
        app.enterEventDispatcher();
    }
    public BrowserFieldDemo()
    {
        pushScreen(new BrowserFieldDemoScreen());
    }
}
class BrowserFieldDemoScreen extends MainScreen
{
    public BrowserFieldDemoScreen()
    {
        BrowserField myBrowserField = new BrowserField();
        add(myBrowserField);
        myBrowserField.requestContent("http://www.blackberry.com");
    }
}
```

Anzeigen von HTML-Inhalt aus einer Ressource in Ihrer Anwendung

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.browser.field2.*;
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `BrowserFieldDemoScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class BrowserFieldDemo extends UiApplication
{
     public static void main(String[] args)
     {
          BrowserFieldDemo app = new BrowserFieldDemo();
          app.enterEventDispatcher();
     }
     public BrowserFieldDemo()
     {
          pushScreen(new BrowserFieldDemoScreen());
     }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm durch Erweitern der `MainScreen`-Klasse.

```
class BrowserFieldDemoScreen extends MainScreen
{
     public BrowserFieldDemoScreen()
     {
     }
}
```

4. Erstellen Sie im Bildschirmkonstruktor eine Instanz der `BrowserField`-Klasse.

   ```
   BrowserField myBrowserField = new BrowserField();
   ```

5. Rufen Sie im Bildschirmkonstruktor `add()` auf, um dem Bildschirm das `BrowserField`-Objekt hinzuzufügen.

   ```
   add(myBrowserField);
   ```

6. Rufen Sie im Bildschirmkonstruktor `BrowserField.requestContent()` auf, um den Ort der Ressource in Ihrer Anwendung anzugeben und den HTML-Inhalt anzuzeigen.

   ```
   myBrowserField.requestContent("local:///test.html");
   ```

   Anmerkung: Die BrowserField-Klasse greift nicht mithilfe einer Ordnerstruktur auf Ressourcen zu. Die `BrowserField`-Klasse zeigt die erste Ressource an, die gefunden wird und die dem angegebenen Dateinamen entspricht.

Codebeispiel: Anzeigen des HTML-Inhalts von einer Ressource in Ihrer Anwendung

```
import net.rim.device.api.browser.field2.*;
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
public class BrowserFieldDemo extends UiApplication
{
    public static void main(String[] args)
    {
        BrowserFieldDemo app = new BrowserFieldDemo();
        app.enterEventDispatcher();
    }
    public BrowserFieldDemo()
    {
        pushScreen(new BrowserFieldDemoScreen());
    }
}
class BrowserFieldDemoScreen extends MainScreen
{
    public BrowserFieldDemoScreen()
    {
        BrowserField myBrowserField = new BrowserField();
        add(myBrowserField);
        myBrowserField.requestContent("local:///test.html");
    }
}
```

Konfigurieren eines Browserfeldes

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.browser.field2.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `BrowserFieldDemoScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class BrowserFieldDemo extends UiApplication
{
     public static void main(String[] args)
     {
          BrowserFieldDemo app = new BrowserFieldDemo();
          app.enterEventDispatcher();
     }
     public BrowserFieldDemo()
     {
          pushScreen(new BrowserFieldDemoScreen());
     }
}
```

3. Erstellen Sie das Framework für den benutzerdefinierten Bildschirm durch Erweitern der `MainScreen`-Klasse.

```
class BrowserFieldDemoScreen extends MainScreen
{
     public BrowserFieldDemoScreen()
     {
     }
}
```

4. Erstellen Sie im Bildschirmkonstruktor eine Instanz der `BrowserFieldConfig`-Klasse.

```
BrowserFieldConfig myBrowserFieldConfig = new BrowserFieldConfig();
```

5. Rufen Sie im Bildschirmkonstruktor `BrowserFieldConfig.setProperty()` auf, um eine Eigenschaft des `BrowserField`-Objekts anzugeben. Der erste Parameter in `setProperty()` gibt die Eigenschaft und der zweite Parameter den Wert der Eigenschaft an. Im folgenden Codebeispiel wird zum Beispiel die `NAVIGATION_MODE`-Eigenschaft eines `BrowserField`-Objekts angegeben:

```
myBrowserFieldConfig.setProperty(BrowserFieldConfig.NAVIGATION_MODE,
BrowserFieldConfig.NAVIGATION_MODE_POINTER);
```

6. Erstellen Sie im Bildschirmkonstruktor eine Instanz der `BrowserField`-Klasse, die die von Ihnen definierte Konfiguration verwendet.

```
BrowserField browserField = new BrowserField(myBrowserFieldConfig);
```

Codebeispiel: Konfigurieren eines Browserfelds

```
import net.rim.device.api.browser.field2.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
public class BrowserFieldDemo extends UiApplication
{
    public static void main(String[] args)
    {
        BrowserFieldDemo app = new BrowserFieldDemo();
        app.enterEventDispatcher();
    }
    public BrowserFieldDemo()
    {
        pushScreen(new BrowserFieldDemoScreen());
    }
}
class BrowserFieldDemoScreen extends MainScreen
{
    public BrowserFieldDemoScreen()
    {
        BrowserFieldConfig myBrowserFieldConfig = new BrowserFieldConfig();
        myBrowserFieldConfig.setProperty(BrowserFieldConfig
    NAVIGATION_MODE,BrowserFieldConfig.NAVIGATION_MODE_POINTER);
        BrowserField browserField = new BrowserField(myBrowserFieldConfig);
    }
}
```

Senden von Formulardaten an eine Webadresse in einem Browserfeld

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.browser.field2.*;
import net.rim.device.api.io.http.*;
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
import java.lang.*;
import java.util.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `BrowserFieldDemoScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class BrowserFieldDemo extends UiApplication
{
     public static void main(String[] args)
     {
          BrowserFieldDemo app = new BrowserFieldDemo();
          app.enterEventDispatcher();
     }
     public BrowserFieldDemo()
     {
          pushScreen(new BrowserFieldDemoScreen());
     }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm durch Erweitern der `MainScreen`-Klasse.

```
class BrowserFieldDemoScreen extends MainScreen
{
     public BrowserFieldDemoScreen()
     {
     }
}
```

4. Erstellen Sie im Bildschirmkonstruktor eine Instanz der `BrowserField`-Klasse.

```
BrowserField browserField = new BrowserField();
```

5. Rufen Sie im Bildschirmkonstruktor `add()` auf, um dem Bildschirm das `BrowserField`-Objekt hinzuzufügen.

```
add(browserField);
```

6. Erstellen Sie im Bildschirmkonstruktor ein `String`-Objekt, das die Basis-Webadresse der Webseite enthält, an die Sie die Formulardaten senden.

```
String baseURL = "http://www.blackberry.com";
```

7. Erstellen Sie im Bildschirmkonstruktor ein `String`-Objekt, das die Formulardaten angibt, die Ihre Anwendung an die Webseite sendet.

```
String postData = "fieldname1=value1&fieldname2=value2";
```

8. Erstellen Sie im Bildschirmkonstruktor ein `Hashtable`-Objekt, um die Kopfzeileninformationen für die Formulardaten zu speichern.

```
Hashtable header = new Hashtable();
```

9. Rufen Sie im Bildschirmkonstruktor `Hashtable.put()` auf, um die Kopfzeileninformationen für die Formulardaten anzugeben.

```
header.put("Content-Length", "" + postData.length());
header.put("Content-Type", "application/x-www-form-urlencoded");
```

10. Rufen Sie im Bildschirmkonstruktor `BrowserField.requestContent()` auf, um die Formulardaten an die Webseite zu senden und die Webseite anzuzeigen.

```
browserField.requestContent(baseURL, postData.getBytes(), new
    HttpHeaders(header));
```

Codebeispiel: Senden von Formulardaten an eine Webadresse in einem Browserfeld

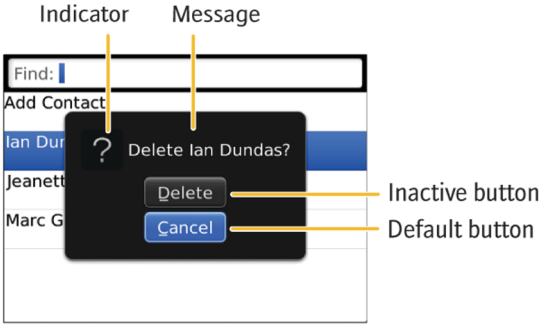
```
import net.rim.device.api.browser.field2.*;
import net.rim.device.api.io.http.*;
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
import java.lang.*;
import java.util.*;
public class BrowserFieldDemo extends UiApplication
{
    public static void main(String[] args)
    {
        BrowserFieldDemo app = new BrowserFieldDemo();
        app.enterEventDispatcher();
    }
    public BrowserFieldDemo()
    {
        pushScreen(new BrowserFieldDemoScreen());
    }
}
class BrowserFieldDemoScreen extends MainScreen
{
    public BrowserFieldDemoScreen()
    {
        BrowserField browserField = new BrowserField();}
        add(browserField);
        String baseURL = "http://www.blackberry.com";
        String postData = "fieldname1=value1&fieldname2=value2";
        Hashtable header = new Hashtable();
        header.put("Content-Length", "" + postData.length());
        header.put("Content-Type", "application/x-www-form-urlencoded");
        browserField.requestContent(baseURL, postData.getBytes(), new
    HttpHeaders(header));
    }
}
```

Dialogfelder

Verwendung von Dialogfeldern, um die folgenden Aktionen auszuführen:

- Fordern Sie von Benutzern Informationen, die erforderlich sind, um eine vom Benutzer eingeleitete Aufgabe abzuschließen.
- Informieren Sie Benutzer über wichtige Informationen oder den Status wichtiger Aktionen.
- Warnen Sie Benutzer vor unerwarteter oder möglicherweise zerstörender Bedingungen oder Situationen.

Dialogfelder sind modal; sie unterbrechen den normalen Vorgang des BlackBerry®-Geräts und werden im Stapel nach oben geschoben. Ein Dialogfeld enthält einen Anzeiger, der die Art des Dialogfelds, eine Nachricht und Schaltflächen anzeigt, die Benutzern ermöglichen, eine Aktion auszuführen. Die Größe des Dialogfelds hängt von der Größe des BlackBerry-Gerätebildschirms ab. Bildlaufpfeile werden falls notwendig angezeigt. Das Thema, das Benutzer auf ihrem BlackBerry-Gerät auswählen, bestimmt den visuellen Stil des Dialogfelds.



Erstellen eines Dialogfelds

Verwenden Sie Alarmdialogfelder, um Benutzer auf eine kritische Aktion hinzuweisen, wie z. B. das Ausschalten des BlackBerry®-Geräts, oder auf einen Fehler, wie z. B. das Eingeben ungültiger Informationen. In einem Alarmdialogfeld wird ein Ausrufezeichen (!) angezeigt. Um ein Alarmdialogfeld zu schließen, können Benutzer auf "OK" klicken oder die Esc-Taste drücken. Weitere Informationen zu anderen Arten von Dialogfeldern finden Sie in der API-Referenz für die BlackBerry® Java® Development Environment.

1. Importieren Sie die `net.rim.device.api.ui.component.Dialog`-Klasse.
2. Erstellen Sie ein Alarmdialogfeld, das den Alarmtext enthält, den Sie anzeigen möchten.

```
Dialog.alert("Specify the alert text that you want to display.")
```

Bewährtes Verfahren: Implementieren von Dialogfeldern

- Verwenden Sie immer Schaltflächen zur Steuerung in Dialogfeldern. Vermeiden Sie die Verwendung von Links oder anderen Steuerelementen.
- Verwenden Sie eine Standardanzeige, die der Art des Dialogfelds entspricht. Vermeiden Sie die Verwendung von Mehrfachanzeigen in einem Dialogfeld.
- Fügen Sie Bildlaufpfeile hinzu, wenn Ihre Dialogfeldnachricht oder Schaltflächen im Dialogfeld nicht vollständig angezeigt werden können. Wenn Sie Standardkomponenten verwenden, werden Bildlaufpfeile bei Bedarf automatisch angezeigt.
- Geben Sie Benutzern immer die Möglichkeit, ein Dialogfeld mit der Esc-Taste zu schließen. Vermeiden Sie das Implementieren einer anderen Aktion, wenn Benutzer zum Schließen eines Dialogfelds die Esc-Taste drücken. Wenn ein Dialogfeld Benutzern beispielsweise die Möglichkeit gibt, eine Einstellung zu ändern, implementieren Sie keine Änderungen, wenn Benutzer die Esc-Taste drücken. Zeigen Sie bei Bedarf das Dialogfeld zu einem späteren Zeitpunkt an.

- Wenn ein Dialogfeld auf einem Anwendungsbildschirm anzeigt wird und Benutzer den Ein-/Aus-Schalter drücken, zeigen Sie die Startseite oder die Anwendungsliste an. Wenn Benutzer zur Anwendung zurückkehren, zeigen Sie wieder das Dialogfeld an.
- Übernehmen Sie wo möglich den visuellen Stil des von den Benutzern ausgewählten Themas. Wenn Ihre Anwendung den visuellen Stil des Themas nicht übernimmt, versuchen Sie den Benutzern eine konstante Arbeitsumgebung bereitzustellen.
- Wählen Sie zum Markieren eine Farbe aus, die einen hohen Kontrast hat und einen deutlichen Fokus impliziert (zum Beispiel weißer Text auf blauem Hintergrund). Verwenden Sie die gleiche Markierungsfarbe für alle Steuerungen.
- Vermeiden Sie eine Festcodierung von Farbwerten. Die von Ihnen gewählte Farbe funktioniert möglicherweise nicht mit dem Thema, das Benutzer ausgewählt haben.

Richtlinien für das Layout

- Zentrieren Sie das Dialogfeld auf dem Bildschirm. Wenn Sie Standardkomponenten verwenden, zentriert das BlackBerry®-Gerät automatisch das Dialogfeld.
- Erstellen Sie Dialogfelder, die 90 % oder weniger der Breite bzw. der Höhe des Bildschirms haben. Wenn Sie Standardkomponenten verwenden, berechnet das BlackBerry-Gerät automatisch die entsprechende Größe der Dialogfelder.
- Zentrieren Sie die Dialogfeldanzeige senkrecht zur Dialogfeldnachricht.
- Zeigen Sie Nachrichten rechts von der Anzeige und oberhalb Schaltflächen an.
- Platzieren Sie Schaltflächen für Bestätigungsaktionen zuerst. Platzieren Sie beispielsweise "Speichern" vor "Verwerfen" oder "Abbrechen".
- Zentrieren Sie Schaltflächen in Dialogfeldern horizontal.
- Platzieren Sie Schaltflächen senkrecht in das Dialogfeld. Das senkrechte Layout ermöglicht die Erweiterung von Schaltflächen, um lokalisierte Schaltflächenbezeichnungen anzupassen.

Richtlinien für Nachrichten

- Seien Sie möglichst genau. Falls möglich, verwenden Sie einen kurzen Satz, der deutlich den Grund zum Anzeigen des Dialogfelds und der Aktionen erklärt, die es schließen können.
- Verwenden Sie nach Möglichkeit für Nachrichten vollständige Sätze.
- Verwenden Sie den Wortschatz, den Benutzer verstehen. Verwenden Sie zum Beispiel "Die Datei konnte nicht gespeichert werden, weil die Medienkarte voll ist" statt "Fehler beim Schreiben der Datei auf den Datenträger".
- Verwenden Sie wo möglich positive Sprache, um ein Tadeln der Benutzer zu vermeiden. Schreiben Sie nie Nachrichten, die Benutzer für Fehler oder unerwartete Bedingungen tadeln. Konzentrieren Sie sich stattdessen auf die Aktionen, die Benutzer unternehmen können, um das Problem zu lösen.
- Verwenden Sie bei der Ansprache der Benutzer die zweite Person (Sie, Ihr).
- Verwenden Sie die übliche Groß-/Kleinschreibung für Sätze.
- Vermeiden Sie in Nachrichten die Verwendung von Ausrufezeichen (!).
- Vermeiden Sie die Verwendung von Auslassungspunkten (...) in Nachrichten, es sei denn, Sie zeigen einen Fortschritt an (zum Beispiel, "Bitte warten...").

Richtlinien für Schaltflächen

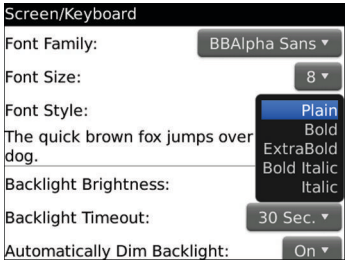
- Verwenden Sie für die Standardschaltfläche jene Schaltfläche, die Benutzer am wahrscheinlichsten anklicken. Vermeiden Sie die Verwendung einer Schaltfläche, die mit einer beendenden Aktion verknüpft ist, als Standardschaltfläche. Eine Ausnahme dieser Regel sind jene Fälle, in denen Benutzer eine kleine beendende Aktion einleiten (wie z. B. Löschen eines einzelnen Elements) und die häufigste Benutzeraktion darin besteht, mit der Aktion fortzufahren.
- Vermeiden Sie die Verwendung von mehr als drei Schaltflächen in einem Dialogfeld. Wenn es mehr als drei gibt, ziehen Sie in Betracht, stattdessen einen Anwendungsbildschirm mit Optionsfeldern zu verwenden.
- Stellen Sie auf BlackBerry-Geräten mit einem Trackwheel, Trackball oder Trackpad Tastenkombinationen für Schaltflächen bereit. In der Regel ist die Tastenkombination der erste Buchstabe der Schaltflächenbezeichnung.
- Verwenden Sie kurze, klare Bezeichnungen.
- Verwenden Sie nach Möglichkeit Bezeichnungen, die aus einem Wort bestehen. Die Größe einer Schaltfläche ändert sich mit der Länge der Bezeichnung. Wenn eine Bezeichnung zu lang ist, werden Auslassungspunkte (...) angezeigt, um zu verdeutlichen, dass der Text abgeschnitten ist. Wenn Sie eine Bezeichnung aus mehreren Wörtern verwenden, verwenden Sie satzabhängige Großschreibung.
- Vermeiden Sie die Verwendung der Bezeichnungen "Ja" und "Nr." Verwenden Sie für Bezeichnungen Verben, die die verknüpfte Aktion beschreiben (zum Beispiel "Abbrechen", "Löschen", "Verwerfen" oder "Speichern"). Dieser Ansatz hilft Benutzern schnell und problemlos zu verstehen, was geschieht, wenn sie auf die Schaltfläche klicken. Falls erforderlich, fügen Sie weitere Beschreibungen anderswo auf dem Bildschirm hinzu (zum Beispiel in einer Anwendungsnachricht).
- Vermeiden Sie die Verwendung von Symbolen oder Grafik in Bezeichnungen.
- Vermeiden Sie Interpunktion in Bezeichnungen. Verwenden Sie in der Bezeichnung einer Schaltfläche Auslassungspunkte um anzuzeigen, dass zusätzliche Informationen erforderlich sind, bevor die verknüpfte Aktion ausgeführt werden kann.

Dropdown-Listen

Verwenden Sie Dropdown-Listen, um einen Satz sich gegenseitig ausschließender Werte bereitzustellen.

Benutzer können mit einer Dropdown-Liste die folgende Aktion ausführen:

Aktion	BlackBerry-Geräte ohne Touchscreen	BlackBerry-Geräte mit Touchscreen
Klicken Sie auf einen Wert aus einer Dropdown-Liste.	Drücken Sie die Leertaste oder klicken Sie auf das Trackwheel, den Trackball oder das Trackpad.	Klicken Sie auf den Bildschirm.



Bewährtes Verfahren: Implementieren von Dropdown-Listen

- Verwenden Sie eine Dropdown-Liste, um zwei oder mehr Auswahlmöglichkeiten anzubieten, wenn der Platz beschränkt ist. Wenn der Platz nicht beschränkt ist, erwägen Sie, stattdessen Optionsschaltflächen zu verwenden, damit die verfügbaren Optionen auf dem Bildschirm angezeigt werden.
- Verwenden Sie die `ObjectChoiceField`-Klasse, um Dropdown-Listen zu erstellen.
- Verwenden Sie für den Standardwert jenen Wert, den Benutzer am wahrscheinlichsten anklicken.
- Verwenden Sie die markierte Option als Standardfokus, wenn Benutzer durch die Liste scrollen.
- Wenn Benutzer nicht auf einen Wert klicken müssen, nehmen Sie einen Wert "Keine" in die Dropdown-Liste auf. Platzieren Sie den Wert "Keine" immer ganz oben in der Liste.
- Gruppieren und ordnen Sie die Werte möglichst logisch (fassen Sie zum Beispiel zusammenhängende Werte in Gruppen zusammen oder beginnen Sie mit den gebräuchlichsten Werten). Vermeiden Sie es, die Werte alphabetisch anzuordnen, weil die alphabetische Reihenfolge sprachspezifisch ist.

Richtlinien für Bezeichnungen

- Verwenden Sie kurze, klare Bezeichnungen für Dropdown-Listen und für die Werte in einer Dropdown-Liste. Vergewissern Sie sich, dass die Bezeichnung klar beschreibt, was geschieht, wenn Benutzer auf den Wert klicken. Die Breite der Dropdown-Liste ändert sich basierend auf der Länge der Wertbezeichnungen. Wenn eine Bezeichnung zu lang ist, werden Auslassungspunkte (...) angezeigt, um zu verdeutlichen, dass der Text abgeschnitten ist.
- Platzieren Sie die Bezeichnung auf der linken Seite einer Dropdown-Liste.
- Verwenden Sie die übliche Groß-/Kleinschreibung für die Bezeichnungen von Dropdown-Listen sowie von Werten in einer Dropdown-Liste.
- Schließen Sie die Bezeichnungen von Dropdown-Listen mit einem Doppelpunkt (:) ab.

Erstellen einer Dropdown-Liste

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `MyUiScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class MyUi extends UiApplication
{
    public static void main(String[] args)
    {
        MyUi theApp = new MyUi();
        theApp.enterEventDispatcher();
    }
```

```
    public MyUi()
    {
        pushScreen(new MyUiScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Bildschirmkonstruktor `setTitle()` auf, um den Titel für den Bildschirm anzugeben.

```
class MyUiScreen extends MainScreen
{
    public MyUiScreen()
    {
        setTitle("UI Component Sample");
    }
}
```

4. Erstellen Sie im Bildschirmkonstruktor mithilfe der `ObjectChoiceField`-Klasse eine Dropdown-Liste, die eine Liste von Wörtern und Formulierungen anzeigt. Erstellen Sie ein `String`-Array, um die Elemente zu speichern, die Sie in der Dropdown-Liste anzeigen möchten. Erstellen Sie ein `int`-Objekt, in dem das Standardelement gespeichert wird, das in der Dropdown-Liste angezeigt werden soll. Geben Sie im `ObjectChoiceField`-Konstruktor die Bezeichnung für die Dropdown-Liste, das Array der anzuzeigenden Elemente und das Standardelement an. Im folgenden Codebeispiel wird standardmäßig "Wednesday" (Mittwoch) angezeigt. Rufen Sie `add()` auf, um die Dropdown-Liste dem Bildschirm hinzuzufügen.

```
String choices[] =
{"Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"};
int iSetTo = 2;
add(new ObjectChoiceField("First Drop-down List",choices,iSetTo));
```

5. Erstellen Sie im Bildschirmkonstruktor mithilfe der `NumericChoiceField`-Klasse eine zweite Dropdown-Liste, die eine Liste von Zahlen anzeigt. Geben Sie im `NumericChoiceField`-Konstruktor die Bezeichnung für die Dropdown-Liste, die erste und die letzte in der Dropdown-Liste anzuzeigende Zahl, die für die Zahlenliste zu verwendende Schrittgröße und die Standardzahl an. Im folgenden Codebeispiel werden die numerischen Parameter in `int`-Objekten gespeichert. In der Dropdown-Liste sind die Zahlen 1 bis 31 enthalten und standardmäßig wird die Zahl 10 angezeigt. Rufen Sie `add()` auf, um die zweite Dropdown-Liste dem Bildschirm hinzuzufügen.

```
int iStartAt   = 1;
int iEndAt     = 31;
int iIncrement = 1;
iSetTo         = 10;
add(new NumericChoiceField("Numeric Drop-Down
    List",iStartAt,iEndAt,iIncrement,iSetTo));
```

6. Um die Standardfunktionalität außer Kraft zu setzen, die den Benutzer auffordert, Änderungen vor dem Schließen der Anwendung zu speichern, setzen Sie in der Erweiterung der `MainScreen`-Klasse die `MainScreen.onSavePrompt()`-Methode außer Kraft. Im folgenden Codebeispiel lautet der Rückgabewert `true`, wodurch angezeigt wird, dass die Anwendung den Benutzer vor dem Schließen nicht auffordert.

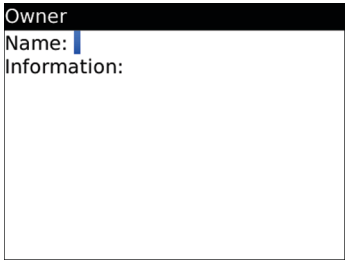
```
public boolean onSavePrompt()
{
    return true;
}
```

Codebeispiel: Erstellen einer Dropdown-Liste

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
public class MyUi extends UiApplication
{
    public static void main(String[] args)
    {
        MyUi theApp = new MyUi();
        theApp.enterEventDispatcher();
    }
    public MyUi()
    {
        pushScreen(new MyUiScreen());
    }
}
class MyUiScreen extends MainScreen
{
    public MyUiScreen()
    {
        setTitle("UI Component Sample");
        String choices[] =
{"Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"};
        int iSetTo = 2;
        add(new ObjectChoiceField("First Drop-down List",choices,iSetTo));
        int iStartAt   = 1;
        int iEndAt     = 31;
        int iIncrement = 1;
        iSetTo         = 10;
        add(new NumericChoiceField("Numeric Drop-Down
List",iStartAt,iEndAt,iIncrement,iSetTo));
    }
    public boolean onSavePrompt()
    {
        return true;
    }
}
```

Bezeichnungen

Verwenden Sie eine Bezeichnung, um Text anzuzeigen, der eine Steuerung kennzeichnet.



Bewährtes Verfahren: Implementieren von Bezeichnungen

- Verwenden Sie die `LabelField`-Klasse, um Bezeichnungen zu erstellen.
- Verwenden Sie kurze, klare Bezeichnungen.
- Gruppieren und ordnen Sie Bezeichnungen logisch an (gruppieren Sie beispielsweise zusammen gehörende Elemente oder führen Sie die häufigsten Elemente zuerst an). Vermeiden Sie es, die Werte alphabetisch anzuordnen, weil die alphabetische Reihenfolge sprachspezifisch ist.
- Verwenden Sie die übliche Groß-/Kleinschreibung für Titel.
- Interpunktieren Sie die Bezeichnung mit einem Doppelpunkt (:).

Erstellen eines Textetiketts

1. Importieren Sie die `net.rim.device.api.ui.component.LabelField`-Klasse.
2. Erstellen Sie eine Instanz von einem `LabelField`, um einem Bildschirm ein Textetikett hinzuzufügen.

```
LabelField title = new LabelField("UI Component Sample", LabelField.ELLIPSIS);
```

Listen

Verwenden Sie eine Liste, um Elemente anzuzeigen, die Benutzer markieren und öffnen können.

Benutzer können in einer Liste die folgenden Aktionen ausführen:

Aktion	BlackBerry-Geräte ohne Touchscreen	BlackBerry-Geräte mit Touchscreen
Markieren Sie ein Element in der Liste.	• Drehen Sie das Trackwheel oder den Trackball. • Fahren Sie mit einem Finger über das Trackpad.	Berühren Sie das Element leicht.

Aktion	BlackBerry-Geräte ohne Touchscreen	BlackBerry-Geräte mit Touchscreen
Öffnen Sie ein Element in der Liste.	Klicken Sie auf das Element oder drücken Sie die Eingabetaste.	Klicken Sie auf den Bildschirm.

Bewährtes Verfahren: Implementieren von Listen

- Verwenden Sie die `ListField`-Klasse, um Listen zu erstellen.

Erstellen eines Listenfelds

Verwenden Sie ein Listenfeld, um eine Liste anzuzeigen, aus der Benutzer eine oder mehrere Werte auswählen können.

1. Importieren Sie die folgenden Klassen:

```
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import java.util.Vector;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die `CreateMenuScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar, der in Schritt 3 beschrieben wird.

```
public class ListFields extends UiApplication
{
     public static void main(String[] args)
     {
         ListFields theApp = new ListFields();
         theApp.enterEventDispatcher();
     }
     public ListFields()
     {
         pushScreen(new ListFieldScreen());
     }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Konstruktor `setTitle()` auf, um den Titel für den Bildschirm anzuzeigen. Rufen Sie `add()` auf, um ein Textfeld auf dem Bildschirm anzuzeigen. Rufen Sie `addMenuItem()` auf, um dem von `MainScreen` erstellten Menü einen Menüeintrag hinzuzufügen.

```
class ListFieldScreen extends MainScreen
{
    private ListField _listField;
```

```
    private Vector _listElements;
    public ListFieldScreen()
    {
        setTitle("List Field Sample");
    }
}
```

4. Erstellen Sie im Bildschirmkonstruktor das Listenfeld. Erstellen Sie ein Array für die Elemente, die Sie dem Listenfeld mithilfe der `Vektors` Klasse hinzufügen möchten. Erstellen Sie das Listenfeld mithilfe der `Listfield()` Klasse. Rufen Sie `add()` auf, um dem Bildschirm das Listenfeld hinzuzufügen. Rufen Sie `initializeList()` auf, die in Schritt 4 beschrieben ist, um ein Listenfeld hinzuzufügen.

```
_listElements = new Vector();
_listField = new ListField();
ListCallback _callback = new ListCallback();
_listField.setCallback(_callback);
add(_listField);
initializeList();
```

5. Erstellen Sie eine Methode, die Elemente anzugeben, die Sie im Listenfeld mithilfe des `String`-Objekts anzeigen wollen. Rufen Sie `addElement()` auf, um die Elemente der Liste hinzuzufügen. Rufen Sie `setSize()` auf, um die Anzahl der Elemente im Listenfeld anzugeben.

```
private void initializeList()
{
    String itemOne = "List item one";
    String itemTwo = "List item two";
    _listElements.addElement(itemOne);
    _listElements.addElement(itemTwo);
    reloadList();
}
private void reloadList()
{
    _listField.setSize(_listElements.size());
}
```

6. Erstellen Sie eine Klasse, welche die `ListFieldCallback`-Schnittstelle implementiert. Implementieren Sie `drawListRow()`, um die Listenfeldelemente zum Bildschirm hinzuzufügen. Implementieren Sie `get()`, um das Listenfeldelement an den angegebenen Index zurückzugeben. Implementieren Sie `indexOfList()`, um das erste Vorkommen einer gegebenen Zeichenfolge zurückzugeben. Implementieren Sie `getPreferredWidth()`, um die Breite des Listenfelds abzurufen.

```
private class ListCallback implements ListFieldCallback
{
    public void drawListRow(ListField list, Graphics g, int index, int y, int w)
    {
        String text = (String)_listElements.elementAt(index);
        g.drawText(text, 0, y, 0, w);
    }
    public Object get(ListField list, int index)
    {
```

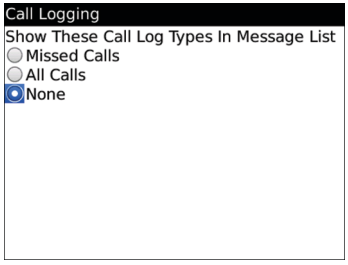
```
        return _listElements.elementAt(index);
    }
    public int indexOfList(ListField list, String prefix, int string)
    {
        return _listElements.indexOf(prefix, string);
    }
    public int getPreferredWidth(ListField list)
    {
        return Display.getWidth();
    }
}
```

Optionsfelder

Verwenden Sie die Optionsfelder, um einen Satz sich gegenseitig ausschließender aber verwandter Auswahlen anzuzeigen.

Benutzer können die folgende Aktion mit den Optionsfeldern ausführen:

Aktion	BlackBerry-Geräte ohne Touchscreen	BlackBerry-Geräte mit Touchscreen
Wählen Sie ein Optionsfeld aus.	Drücken Sie die Leertaste oder klicken Sie auf das Trackwheel, den Trackball oder das Trackpad.	Klicken Sie auf den Bildschirm.



Bewährtes Verfahren: Implementieren von Optionsschaltflächen

- Verwenden Sie Optionsschaltflächen, um zwei oder mehr Auswahlmöglichkeiten anzubieten, wenn der Platz nicht beschränkt ist. Wenn der Platz beschränkt ist, erwägen Sie, stattdessen eine Dropdown-Liste zu verwenden.
- Verwenden Sie die `RadioButtonField`-Klasse, um Optionsschaltflächen zu erstellen.
- Vergewissern Sie sich, dass der Inhalt für Optionsschaltflächen statisch bleibt. Inhalt für Optionsschaltflächen sollte nicht in Abhängigkeit vom Kontext geändert werden.
- Starten Sie keine Aktion, wenn Benutzer eine Optionsschaltfläche auswählen. Zum Beispiel keinen neuen Bildschirm öffnen.
- Richten Sie Optionsschaltflächen vertikal aus.

- Gruppieren und ordnen Sie die Werte möglichst logisch (fassen Sie zum Beispiel zusammenhängende Optionsschaltflächen in Gruppen zusammen oder beginnen Sie mit den gebräuchlichsten Werten). Vermeiden Sie es, die Optionsschaltflächen alphabetisch anzuordnen, weil die alphabetische Reihenfolge sprachspezifisch ist.

Richtlinien für Bezeichnungen

- Verwenden Sie kurze, klare Bezeichnungen. Vergewissern Sie sich, dass das Bezeichnung klar beschreibt, was geschieht, wenn Benutzer die Optionsschaltfläche auswählen. Wenn die Bezeichnungen zu lang sind, erfolgt ein Zeilenumbruch.
- Platzieren Sie Bezeichnungen rechts von den Optionsschaltflächen.
- Verwenden Sie die übliche Groß-/Kleinschreibung für Titel.
- Verwenden Sie keine Ende-Interpunktion.

Erstellen einer Optionstaste

Sie können eine Gruppe der Optionstasten mithilfe der `RadioButtonGroup`-Klasse erstellen. Ein Benutzer kann nur eine Option von einer Gruppe der Optionstasten auswählen.

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `MyUiScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class MyUi extends UiApplication
{
    public static void main(String[] args)
    {
        MyUi theApp = new MyUi();
        theApp.enterEventDispatcher();
    }
    public MyUi()
    {
        pushScreen(new MyUiScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Bildschirmkonstruktor `setTitle()` auf, um den Titel für den Bildschirm anzugeben.

```
class MyUiScreen extends MainScreen
{
    public MyUiScreen()
    {
```

```
        setTitle("UI Component Sample");
    }
}
```

4. Erstellen Sie im Bildschirmkonstruktor eine Gruppe der Optionstasten mithilfe der `RadioButtonGroup`-Klasse. Erstellen Sie die Optionstasten, die Sie der Gruppe mithilfe der `RadioButtonField`-Klasse hinzufügen wollen. Im `RadioButtonField`-Konstruktor geben Sie das Etikett für die Optionstaste, die Gruppe und einen booleschen Wert an, um die Standardauswahl anzuzeigen (zum Beispiel zeigt `wahr` an, dass standardmäßig die Option ausgewählt ist). Rufen Sie `add()` auf, um dem Bildschirm die Optionstasten hinzuzufügen.

```
RadioButtonGroup rbg = new RadioButtonGroup();
add(new RadioButtonField("Option 1",rbg,true));
add(new RadioButtonField("Option 2",rbg,false));
```

5. Um die Standardfunktionalität außer Kraft zu setzen, die den Benutzer auffordert, Änderungen vor dem Schließen der Anwendung zu speichern, setzen Sie in der Erweiterung der `MainScreen`-Klasse die `MainScreen.onSavePrompt()`-Methode außer Kraft. Im folgenden Codebeispiel lautet der Rückgabewert `true`, wodurch angezeigt wird, dass die Anwendung den Benutzer vor dem Schließen nicht auffordert.

```
public boolean onSavePrompt()
{
    return true;
}
```

Codebeispiel: Erstellen eines Optionsfelds

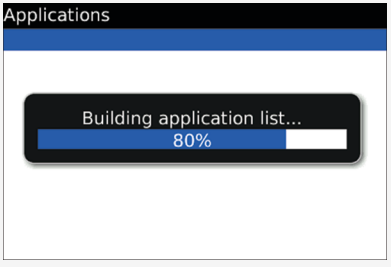
```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
public class MyUi extends UiApplication
{
    public static void main(String[] args)
    {
        MyUi theApp = new MyUi();
        theApp.enterEventDispatcher();
    }
    public MyUi()
    {
        pushScreen(new MyUiScreen());
    }
}
class MyUiScreen extends MainScreen
{
    public MyUiScreen()
    {
        setTitle("UI Component Sample");
        RadioButtonGroup rbg = new RadioButtonGroup();
        add(new RadioButtonField("Option 1",rbg,true));
        add(new RadioButtonField("Option 2",rbg,false));
    }
```

```
    public boolean onSavePrompt()
    {
        return true;
    }
}
```

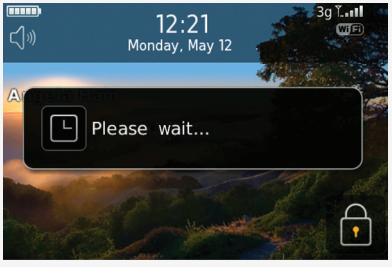
Fortschrittsanzeigen

Verwenden Sie Fortschrittanzeigen, um den Status eines Vorgangs anzuzeigen. Es gibt zwei Arten von Fortschrittanzeigen:

- Verwenden Sie bestimmte Fortschrittsanzeigen, wenn die Zeitdauer eines Vorgangs bekannt ist. Bestimmte Fortschrittanzeigen enthalten ein Etikett, um den Vorgang anzuzeigen und eine horizontale Leiste, die sich von links nach rechts füllt, während ein Vorgang fortschreitet. Ein Prozentsatz wird in der Leiste angezeigt, um anzuzeigen, wie viel vom Vorgang vollständig ist. Im Browser zeigen Fortschrittanzeigen auch die Anzahl der Kilobytes aus einer Gesamtanzahl der Kilobytes an, die ein BlackBerry®-Gerät heruntergeladen hat. Um einen bestimmten Fortschrittanzeiger auszublenden, drücken Benutzer die Taste "Beenden". Wenn Benutzer die Taste "Beenden" drücken, setzt der Vorgang fort, aber Benutzer können andere Aufgaben gleichzeitig ausführen.
- Verwenden Sie eine unbestimmte Fortschrittsanzeige, wenn die Zeitdauer eines Vorgangs unbekannt ist. Sie können den Fortschritt mithilfe eines Statusdialogfeldes oder eines animierten Uhrcursors für das Precision-Thema oder ein Sanduhrsymbol für das Dimension-Thema verwenden,. anzeigen. Benutzer können andere Aktionen nicht ausführen, während eine Anwendung einen unbestimmten Fortschrittanzeiger anzeigt.



Bestimmter Fortschrittanzeiger



Unbestimmter Fortschrittanzeiger

Bewährtes Verfahren: Implementieren von Fortschrittsanzeigen

- Zeigen Sie immer den Fortschritt an, wenn der Abschluss eines Vorgangs mehr als 2 Sekunde dauert.
- Verwenden Sie eine bestimmte Fortschrittsanzeige, wenn die Zeitdauer eines Vorgangs bekannt ist.
- Verwenden Sie eine unbestimmte Fortschrittsanzeige, wenn die Zeitdauer eines Vorgangs unbekannt ist.

- Verwenden Sie die `GaugeField`-Klasse, um Fortschrittsanzeigen zu erstellen.
- Ermöglichen Sie Benutzern immer, eine bestimmte Fortschrittsanzeige mithilfe der Beenden-Taste auszublenden.
- Stellen Sie hilfreiche Fortschrittsinformationen bereit. Wenn Benutzer zum Beispiel eine Webseite laden, zeigen Sie die Datenmenge an, die ihr BlackBerry®-Gerät bereits geladen hat (zum Beispiel 8 KB von 10 KB). Wenn sie Anwendungen auf ihr Gerät herunterladen, zeigen Sie den Fortschritt als Prozentsatz an. Seien Sie bei den Fortschrittsinformationen so genau wie möglich.
- Um den unbestimmten Fortschritt anzuzeigen, verwenden Sie ein Uhrensymbol für das Precision-Thema oder ein Sanduhrsymbol für das Dimension-Thema.
- Verwenden Sie ein Statusdialogfeld, um den unbestimmten Fortschritt anzuzeigen, wenn Sie eine aussagekräftige Meldung über den aktuellen Vorgang bereitstellen können. Um die Garbage Collection anzuzeigen, verwenden Sie den animierten Uhren-Cursor für das Precision-Thema oder einen animierten Sanduhr-Cursor für das Dimension-Thema.
- Übernehmen Sie wo möglich den visuellen Stil des von den Benutzern ausgewählten Themas. Wenn Ihre Anwendung den visuellen Stil des Themas nicht übernimmt, versuchen Sie den Benutzern eine konstante Arbeitsumgebung bereitzustellen.

Richtlinien für Bezeichnungen

- Verwenden Sie eine kurze und klare beschreibende Bezeichnung (zum Beispiel "Daten werden geladen" oder "Anwendungsliste wird erstellt").
- Verwenden Sie die übliche Groß-/Kleinschreibung für Sätze.
- Schließen Sie die Dialogfeldmeldung mit Auslassungspunkten (...) ab.

Erstellen einer Fortschrittsanzeige

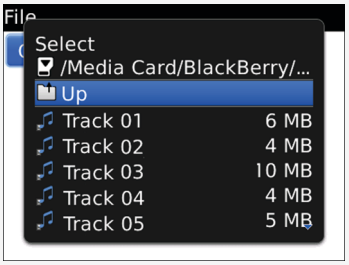
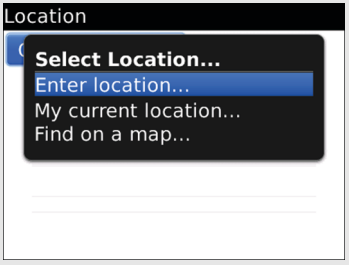
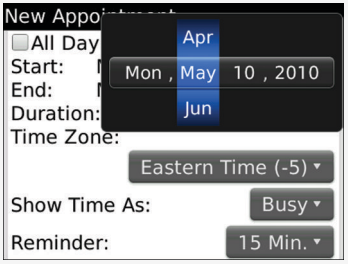
1. Importieren Sie die `net.rim.device.api.ui.component.GaugeField`-Klasse.
2. Erstellen Sie eine Instanz von `GaugeField`.

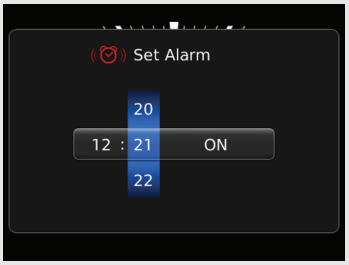
```
GaugeField percentGauge = new GaugeField("Percent: ", 1, 100, 29,
GaugeField.PERCENT);
```

Auswahldialogfelder

Sie können Auswahldialogfelder verwenden, um Benutzern zu helfen, ein Element von einer Liste problemlos auszuwählen.

Auswahltyp	Beschreibung
Datei	Diese Auswahl ermöglicht Benutzern, ihre BlackBerry®-Geräte nach Dateien zu durchsuchen.

Auswahltyp	Beschreibung
	
Ort	Diese Auswahl ermöglicht Benutzern, einen Ort von einer Liste auszuwählen, die Sie definieren. Zum Beispiel können Sie Benutzern ermöglichen, ihre GPS-Position oder einen vorher ausgewählte Ort auszuwählen. 
Date	Diese Auswahl ermöglicht Benutzern, einen bestimmten Tag, Monat oder ein bestimmtes Jahr auszuwählen Zum Beispiel können Sie Benutzern ermöglichen, einen Monat und ein Jahr auszuwählen, um anzugeben, wann ihre Kreditkarte abläuft. 
Time	Diese Auswahl ermöglicht Benutzern, eine bestimmte Stunde, Minute oder Sekunde auszuwählen.

Auswahltyp	Beschreibung
	

Bewährtes Verfahren: Implementieren von Auswahldialogfeldern

Verwenden Sie die Klassen `FilePicker`, `LocationPicker` und `DateTimePicker`, um Auswahldialogfelder zu erstellen.

Richtlinien für Dateiauswahldialogfelder

- Ermöglichen Sie den Benutzern, über einen entsprechenden Standardordner mit dem Durchsuchen zu beginnen. Wenn die Anwendung keinen Standardordner besitzt und eine Medienkarte in das BlackBerry-Gerät eingesetzt wird, ermöglichen Sie den Benutzern, vom Stammordner aus mit dem Durchsuchen zu beginnen.
- Wenn die Anwendung verschiedene Dateitypen unterstützt und eine Medienkarte in das BlackBerry-Gerät eingesetzt wird, ermöglichen Sie den Benutzern, von "/Media Card/BlackBerry" aus mit dem Durchsuchen zu beginnen. Wenn die Anwendung verschiedene Dateitypen unterstützt und keine Medienkarte in das BlackBerry-Gerät eingesetzt wird, ermöglichen Sie den Benutzern, von "/Device Memory/home/user" aus mit dem Durchsuchen zu beginnen.

Erstellen einer Datumsauswahl

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.picker.*;
import java.util.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die `DatePickScreen`-Klasse stellt den in Schritt 3 beschriebenen benutzerdefinierten Bildschirm dar.

```
public class DatePick extends UiApplication
{
    public static void main(String[] args)
    {
```

```
        DatePick theApp = new DatePick();
        theApp.enterEventDispatcher();
    }
    public DatePick()
    {
        pushScreen(new DatePickScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Konstruktor `setTitle()` auf, um einen Titel auf dem Bildschirm anzuzeigen. Rufen Sie `add()` auf, um ein Rich-Text-Feld auf dem Bildschirm anzuzeigen.

```
class DatePickScreen extends MainScreen
{
    public DatePickScreen()
    {
        setTitle("Date Picker Sample");
        add(new RichTextField("Trying Date Picker"));
    }
}
```

4. Fügen Sie der Ereigniswarteschlange der Anwendung durch Aufrufen von `invokeLater()` einen Codeabschnitt hinzu. Erstellen Sie ein `Runnable`-Objekt und übergeben Sie es als Parameter an `invokeLater()`. Setzen Sie `run()` in der Definition von `Runnable` außer Kraft.

```
class DatePickScreen extends MainScreen
{
    public DatePickScreen()
    {
        setTitle("Date Picker Sample");
        add(new RichTextField("Trying Date Picker"));
        UiApplication.getUiApplication().invokeLater(new Runnable()
        {
            public void run()
            {
            }
        });
    }
}
```

5. Rufen Sie in `run()` die Methode `DateTimePicker.getInstance()` auf, um ein `DateTimePicker`-Objekt zurückzugeben. Rufen Sie `doModal()` auf, um die Datumsauswahl anzuzeigen. Rufen Sie `doModal()` auf, um ein `Calendar`-Objekt zurückzugeben, das das Datum und die Uhrzeit darstellt, die der Benutzer auswählt. Verwenden Sie `getTime()`, um das Datum und die Uhrzeit als `Date`-Objekt zurückzugeben. Verwenden Sie `Dialog.alert()`, um das ausgewählte Datum und die Uhrzeit anzuzeigen.

```
class DatePickScreen extends MainScreen
{
    public DatePickScreen()
    {
```

```
        setTitle("Date Picker Sample");
        add(new RichTextField("Trying Date Picker"));
        UiApplication.getUiApplication().invokeLater(new Runnable()
        {
            public void run()
            {
                DateTimePicker datePicker = DateTimePicker.getInstance();
                datePicker.doModal();
                Calendar cal = datePicker.getDateTime();
                Date date = cal.getTime();
                Dialog.alert("You selected " + date.toString());
            }
        });
    }
}
```

Codebeispiel: Erstellen einer Datumsauswahl

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.picker.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.database.*;
import net.rim.device.api.io.*;
import java.util.*;
public class DatePick extends UiApplication
{
    public static void main(String[] args)
    {
        DatePick theApp = new DatePick();
        theApp.enterEventDispatcher();
    }
    public DatePick()
    {
        pushScreen(new DatePickScreen());
    }
}
class DatePickScreen extends MainScreen
{
    public DatePickScreen()
    {
        setTitle("Date Picker Sample");
        add(new RichTextField("Trying Date Picker"));
        UiApplication.getUiApplication().invokeLater(new Runnable()
        {
            public void run()
            {
                DateTimePicker datePicker = DateTimePicker.getInstance();
                datePicker.doModal();
                Calendar cal = datePicker.getDateTime();
                Date date = cal.getTime();
```

```
                Dialog.alert("You selected " + date.toString());
            }
        });
    }
}
```

Erstellen einer Dateiauswahl

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.picker.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import java.util.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die `DatePickScreen`-Klasse stellt den in Schritt 3 beschriebenen benutzerdefinierten Bildschirm dar.

```
public class FilePick extends UiApplication
{
    public static void main(String[] args)
    {
        FilePick theApp = new FilePick();
        theApp.enterEventDispatcher();
    }
    public FilePick()
    {
        pushScreen(new FilePickScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Bildschirmkonstruktor `setTitle()` auf, um einen Titel für den Bildschirm anzugeben. Rufen Sie `add()` auf, um dem Bildschirm ein Bezeichnungsfeld hinzuzufügen.

```
class FilePickScreen extends MainScreen
{
    public FilePickScreen()
    {
        setTitle("File Picker Sample");
        add(new LabelField("Trying File Picker"));
    }
}
```

4. Rufen Sie im Bildschirmkonstruktor `invokeLater()` auf, um der Ereigniswarteschlange der Anwendung einen Codeabschnitt hinzuzufügen. Erstellen Sie ein `Runnable`-Objekt und übergeben Sie es als Parameter an `invokeLater()`.

```
class FilePickScreen extends MainScreen
{
    public FilePickScreen()
    {
        setTitle("File Picker Sample");
        add(new LabelField("Trying File Picker"));
        UiApplication.getUiApplication().invokeLater(new Runnable()
        {
            public void run()
            {
            }
        });
    }
}
```

5. Setzen Sie `run()` in der Definition von `Runnable` außer Kraft. Rufen Sie in `run()` die Methode `FilePicker.getInstance()` auf, um ein `FilePicker`-Objekt zurückzugeben. Erstellen Sie ein `FilePickListener`-Objekt, das in Schritt 6 beschrieben wird. Rufen Sie `setListener()` auf, um den Listener für die Dateiauswahl zu registrieren. Rufen Sie `show()` auf, um die Dateiauswahl auf dem Bildschirm anzuzeigen.

```
class FilePickScreen extends MainScreen
{
    public FilePickScreen()
    {
        setTitle("File Picker Sample");
        add(new LabelField("Trying File Picker"));
        UiApplication.getUiApplication().invokeLater(new Runnable()
        {
            public void run()
            {
                FilePicker fp = FilePicker.getInstance();
                FilePickListener fileListener = new FilePickListener();
                fp.setListener(fileListener);
                fp.show();
            }
        });
    }
}
```

6. Rufen Sie `Dialog.alert` auf, um ein Dialogfeld mit einer Meldung zu erstellen, die anzeigt, welche Datei ausgewählt wurde. Rufen Sie `Dialog.alert` in einer Klasse auf, die die `FilePicker.Listener`-Schnittstelle durch Außerkraftsetzen von `selectionDone()` implementiert. Die Anwendung ruft `selectionDone()` auf, wenn der Benutzer mithilfe der Dateiauswahl eine Datei auswählt.

```
class FilePickListener implements FilePicker.Listener
{
    public void selectionDone(String str)
    {
        Dialog.alert("You selected " + str);
    }
}
```

Codebeispiel: Erstellen eines Dateipickers

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.picker.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.io.*;
public class FilePick extends UiApplication
{
    public static void main(String[] args)
    {
        FilePick theApp = new FilePick();
        theApp.enterEventDispatcher();
    }
    public FilePick()
    {
        pushScreen(new FilePickScreen());
    }
}
class FilePickScreen extends MainScreen
{
    public FilePickScreen()
    {
        setTitle("File Picker Sample");
        add(new LabelField("Trying File Picker"));
        UiApplication.getUiApplication().invokeLater(new Runnable()
        {
            public void run()
            {
                FilePicker fp = FilePicker.getInstance();
                FilePickListener fileListener = new FilePickListener();
                fp.setListener(fileListener);
                fp.show();
            }
        });
    }
}
class FilePickListener implements FilePicker.Listener
{
    public void selectionDone(String str)
    {
```

```
            Dialog.alert("You selected " + str);
        }
}
```

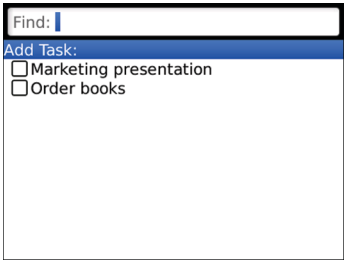
Suchfelder

Setzen Sie Suchfelder ein, um es Benutzern zu ermöglichen, nach Elementen in einer Anwendung zu suchen. Zahlreiche BlackBerry®-Geräteanwendungen enthalten Suchfelder. Zum Beispiel können Benutzer in der Aufgabenliste ein Suchfeld verwenden, um nach bestimmten Aufgaben zu suchen.

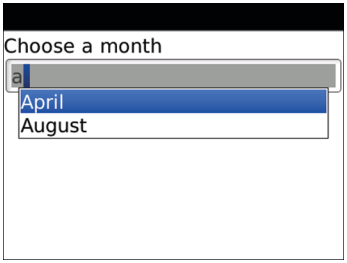
Während Benutzer Text in ein Suchfeld eingeben, sucht die Anwendung danach und zeigt Elemente an, die mit dem Suchtext beginnen. Wenn Benutzer mehrere Wörter in das Suchfeld eingeben, zeigt die Anwendung Ergebnisse an, die beiden Wörtern entsprechen. Wenn Benutzer beispielsweise "bi zu" eingeben, gibt die Anwendung alle Elemente zurück, die ein Wort enthalten, das mit "bi" beginnt und ein Wort enthalten, das mit "zu" beginnt (wie z. B. "bitte zurückrufen").

Benutzer können in einem Suchfeld folgende Aktionen ausführen:

Aktion	BlackBerry-Geräte ohne Touchscreen	BlackBerry-Geräte mit Touchscreen
Öffnen Sie ein markiertes Element in den Suchergebnissen.	Drücken Sie die Eingabetaste oder klicken Sie auf das Trackwheel, Trackpad oder den Trackball.	Klicken Sie auf den Bildschirm.
Löschen Sie den Text in einem Suchfeld.	Drücken Sie die Esc-Taste.	Drücken Sie die Esc-Taste.



Um Benutzern zu helfen, einen bestimmten Wert schnell einzugeben, können Sie ein Feld erstellen, das automatische Vervollständigung unterstützt. Beim Eingeben erhalten Benutzer eine Dropdown-Liste mit vorgeschlagenen Werten angezeigt. Benutzer können auf einen dieser Werte klicken oder die Eingabe fortsetzen, um die Werte in der Liste einzuschränken.



Bewährtes Verfahren: Implementieren von Suchfeldern

- Stellen Sie ein Suchfeld bereit, wenn Sie eine erzwungene Liste von Elementen haben.
- Verwenden Sie die `KeywordFilterField`-Klasse, um Suchfelder zu erstellen.
- Verwenden Sie die `AutoCompleteField`-Klasse und die `BasicFilteredList`-Klasse, um autocomplete-Felder zu erstellen.
- Geben Sie Benutzern die Möglichkeit, von ihnen eingegebenen Suchtext mit der Esc-Taste zu löschen.
- Setzen Sie den Standardfokus auf das Suchfeld und das erste Element in der Liste.
- Zeigen Sie Elemente in den Suchergebnissen an, die ein Wort enthalten, das mit dem eingegebenen Suchentext beginnt.
- Weisen Sie einem Bildschirm, der ein Suchfeld enthält, keine Tastenkombinationen zu. Wenn Sie Tastenkombinationen verwenden wollen, und Sie haben eine erzwungene Liste von Elementen, stellen Sie alternative Suchfunktionen bereit. Zum Beispiel könnten Benutzer nach Nachrichten in einer Nachrichtenliste durch Drücken von S oder Verwenden des Menüs suchen.
- Platzieren Sie ein Suchfeld in die Titelleiste eines Anwendungsbildschirms.
- Machen Sie Suchfelder nicht von Groß- und Kleinschreibung abhängig.

Richtlinien für Bezeichnungen

- Verwenden Sie für das Suchfeld die Bezeichnung "Suchen:".
- Das Suchfeld darf keinen Eingabeaufforderungstext enthalten.

Erstellen eines Suchfelds

Sie können eine Anwendung erstellen, die die `KeywordFilterField`-Klasse verwendet, die im `net.rim.device.api.ui.component`-Paket enthalten ist, um ein Benutzeroberflächenfeld bereitzustellen, das aus einem einzelnen Texteingabefeld und einer Liste auswählbarer Elemente besteht. Während die Benutzer Text in ein Suchfeld eingeben, filtert die Anwendung die Elemente in der Liste, die mit dem Suchtext beginnen. Weitere Informationen zur Verwendung der `KeywordFilterField`-Klasse finden Sie in der Beispielanwendung zum Stichwortfilterfeld, die in der BlackBerry® Java® Development Environment Version 4.3.1 oder höher enthalten ist.

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.collection.util.SortedReadableList;
import net.rim.device.api.io.LineReader;
import net.rim.device.api.ui.component.KeywordFilterField;
```

```
import net.rim.device.api.ui.component.KeywordProvider;
import java.io.InputStream;
import java.lang.String;
import java.util.Vector;
```

2. Erstellen Sie Variablen. Im folgenden Codebeispiel erweitert `CountryList` die `SortedReadableList`-Klasse und implementiert die `KeywordProvider`-Schnittstelle.

```
private KeywordFilterField _keywordField;
private CountryList _CountryList;
private Vector _countries;
```

3. Um eine Liste auswählbarer Textelemente zu erstellen, füllen Sie einen Vektor mit Daten aus einer Textdatei.

```
_countries = getDataFromFile();
```

4. Erstellen Sie eine Instanz einer Klasse, die die `SortedReadableList`-Klasse erweitert.

```
_CountryList = new
    CountryList(StringComparator.getInstance(true),_countries);
```

5. Um die Elemente der Liste anzugeben, erstellen Sie eine neue Instanz eines `KeywordFilterField`-Objekts.

```
_keywordField = new KeywordFilterField();
```

6. Rufen Sie `KeywordFilterField.setlist()` auf.

```
_keywordField.setList(_CountryList, _CountryList);
```

7. Legen Sie eine Bezeichnung für das Eingabefeld von `KeywordFilterField` fest.

```
_keywordField.setLabel("Search: ");
```

8. Erstellen Sie den Hauptbildschirm der Anwendung und fügen Sie dem Hauptbildschirm ein `KeywordFilterField`-Objekt hinzu.

```
KeywordFilterDemoScreen screen = new
    KeywordFilterDemoScreen(this,_keywordField);
screen.add(_keywordField.getKeywordField());
screen.add(_keywordField);
pushScreen(screen);
```

9. Um eine Methode zu erstellen, die einen Vektor von `Country`-Objekten füllt und zurückgibt, die Daten aus der Textdatei enthalten, geben Sie in der Methodensignatur als Rückgabetyp `Vector` an.

```
public Vector getDataFromFile()
```

10. Erstellen und speichern Sie einen Verweis auf ein neues `Vector`-Objekt.

```
Vector countries = new Vector();
```

11. Erstellen Sie einen Eingabestrom zur Textdatei.

```
InputStream stream =
    getClass().getResourceAsStream("/Data/CountryData.txt");
```

12. Lesen Sie durch CRLF getrennte Zeilen aus dem Eingabestrom.

```
LineReader lineReader = new LineReader(stream);
```

13. Lesen Sie Zeile für Zeile Daten aus dem Eingabestrom, bis Sie die Markierung für das Dateiende erreichen. Jede Zeile wird analysiert, um Daten zu extrahieren, die für das Erstellen von `Country`-Objekten verwendet werden.

```
for(;;){
    //Obtain a line of text from the text file
    String line = new String(lineReader.readLine());
    //If we are not at the end of the file, parse the line of text
    if(!line.equals("EOF")) {
        int space1 = line.indexOf(" ");
        String country = line.substring(0,space1);
        int space2 = line.indexOf(" ",space1+1);
        String population = line.substring(space1+1,space2);
        String capital = line.substring(space2+1,line.length());
        // Create a new Country object
       countries.addElement(new Country(country,population,capital));

    }
    else {
        break;
    }
} // end the for loop
return countries;
```

14. Um der Liste auswählbarer Textelemente ein Stichwort hinzuzufügen, rufen Sie `SortedReadableList.doAdd(element)` auf.

```
SortedReadableList.doAdd(((Country)countries.elementAt(i))
    getCountryName()) ;
```

15. Um die Liste auswählbarer Textelemente zu aktualisieren, rufen Sie `KeywordFilterField.updatelist()` auf.

```
_keywordField.updateList();
```

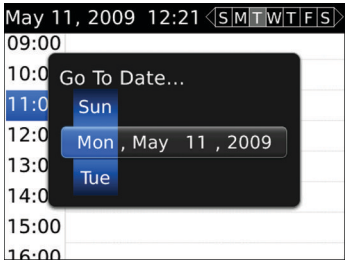
16. Um das Stichwort zu erhalten, das ein BlackBerry-Gerätebenutzer in das `KeywordFilterField`-Feld eingegeben hat, rufen Sie `KeywordFilterField.getKeyword()` auf.

```
String userTypedWord = _keywordField.getKeyword();
```

Drehungsfelder

Verwenden Sie ein Drehungsfeld, um Benutzern die problemlose Auswahl eines Elements aus einer sortierten Liste zu ermöglichen. Verwenden Sie beispielsweise Drehungsfelder, um es Benutzern zu ermöglichen, eine Zahl zu finden oder den Wochentag zu ändern.

Aktion	BlackBerry-Geräte ohne Touchscreen	BlackBerry-Geräte mit Touchscreen
Finden Sie ein Element in der Liste.	• Drehen Sie den Trackball auf- oder abwärts. • Schieben Sie einen Finger auf dem Trackpad nach oben oder unten.	Fahren Sie mit einem Finger schnell nach oben oder unten.
Wählen Sie ein Element aus der Liste aus.	Klicken Sie auf den Trackball oder mit dem Trackpad.	Klicken Sie auf OK.
Gehen Sie zu einem anderen Drehungsfeld.	• Drehen Sie den Trackball nach links oder rechts. • Fahren Sie mit einem Finger auf dem Trackpad nach links oder rechts.	Berühren Sie das Drehungsfeld.



Bewährtes Verfahren: Implementieren von Drehfeldern

- Verwenden Sie Drehfelder bei einer Liste sequenzieller Elemente.
- Verwenden Sie Dropdown-Listen bei nicht sequenziellen Elementen oder Elementen mit unregelmäßigen Abständen. Bei einer kurzen Liste nicht sequenzieller Elemente können Sie ein Drehfeld verwenden, um eine interaktivere Funktionalität für Benutzer bereitzustellen.
- Vermeiden Sie die Verwendung eines Drehfelds, wenn mehrere andere Steuerelemente auf dem Bildschirm angezeigt werden.
- Verwenden Sie die Klassen `SpinBoxField` und `SpinBoxFieldManager`, um Drehfelder zu erstellen.
- Fügen Sie Dialogfelder wenn möglich Drehfeldern statt Bildschirmen hinzu. Auf BlackBerry-Geräten mit einem Touchscreen müssen Benutzer auf "OK" klicken, um ein Element aus der Liste auszuwählen.
- Wenn Benutzer ein Drehfeld markieren, zeigen Sie drei bis fünf Elemente übereinander an.
- Verwenden Sie ein erkennbares Muster für die Abfolge der Elemente (zum Beispiel 5, 10, 15), damit Benutzer abschätzen können, wie weit sie scrollen müssen, um das Zielelement zu finden.
- Vermeiden Sie, Benutzer horizontal scrollen zu lassen, um mehrere Drehfelder anzuzeigen. Unterteilen Sie Drehfelder falls notwendig in mehrere Felder.

- Wenn der Text in einem Drehfeld zu lang ist, verwenden Sie Auslassungspunkte (...).

Erstellen eines Drehfelds

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.UiApplication;
import net.rim.device.api.ui.component.Dialog;
import net.rim.device.api.ui.component.TextSpinBoxField;
import net.rim.device.api.ui.container.MainScreen;
import net.rim.device.api.ui.container.SpinBoxFieldManager;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die `HomeScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar, der in Schritt 3 beschrieben wird.

```
public class SpinBoxApp extends UiApplication
{
    public static void main(String[] args)
    {
        SpinBoxApp app = new SpinBoxApp();
        app.enterEventDispatcher();
    }
    public SpinBoxApp()
    {
        pushScreen(new HomeScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Deklarieren Sie eine Variable für jedes Feld im Drehfeld sowie eine Variable für den Drehfeld-Manager.

```
class HomeScreen extends MainScreen
{
    TextSpinBoxField spinBoxDays;
       TextSpinBoxField spinBoxMonths;
       SpinBoxFieldManager spinBoxMgr;
    public HomeScreen()
    {
    }
}
```

4. Erstellen Sie im Bildschirmkonstruktor für jedes der Drehfelder ein Array von String-Objekten.

```
final String[] DAYS   =
    {"Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"};
final String[] MONTHS =
{"January","February","March","April","May","June","July","Au
    ust","September","October","November","December"};
```

5. Erstellen Sie im Bildschirmkonstruktor neue Instanzen des Drehfeld-Managers sowie der zwei Drehfelder. Übergeben Sie das entsprechende Array von Zeichenfolgen als Argumente an jeden der Drehfeldkonstruktoren.

```
spinBoxMgr = new SpinBoxFieldManager();
spinBoxDays   = new TextSpinBoxField(DAYS);
spinBoxMonths = new TextSpinBoxField(MONTHS);
```

6. Fügen Sie im Bildschirmkonstruktor dem Drehfeld-Manager die Drehfelder hinzu. Rufen Sie `add()` auf, um den Manager sowie die Felder, die er enthält, dem Bildschirm hinzuzufügen.

```
spinBoxMgr.add(spinBoxDays);
spinBoxMgr.add(spinBoxMonths);
add(spinBoxMgr);
```

Codebeispiel: Erstellen eines Drehfelds

```
import net.rim.device.api.ui.UiApplication;
import net.rim.device.api.ui.container.MainScreen;
import net.rim.device.api.ui.container.SpinBoxFieldManager;
import net.rim.device.api.ui.component.Dialog;
import net.rim.device.api.ui.component.TextSpinBoxField;
public class SpinBoxApp extends UiApplication
{
    public static void main(String[] args)
       {
              SpinBoxApp app = new SpinBoxApp();
              app.enterEventDispatcher();
    }
       public SpinBoxApp()
       {
              HomeScreen homeScreen = new HomeScreen();
              pushScreen(homeScreen);
       }
}
class HomeScreen extends MainScreen
{
    TextSpinBoxField spinBoxDays;
       TextSpinBoxField spinBoxMonths;
       SpinBoxFieldManager spinBoxMgr;
       public HomeScreen()
       {
           final String[] DAYS   =
        {"Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"};
              final String[] MONTHS =
            {"January","February","March","April","May","June","July","August","Septe
            ber","October","November","December"};
           spinBoxMgr = new SpinBoxFieldManager();
           spinBoxDays   = new TextSpinBoxField(DAYS);
              spinBoxMonths = new TextSpinBoxField(MONTHS);
              spinBoxMgr.add(spinBoxDays);
```

```
                spinBoxMgr.add(spinBoxMonths);
                add(spinBoxMgr);
        }
        public void close()
        {
                Dialog.alert("You selected " +
              (String)spinBoxDays.get(spinBoxDays.getSelectedIndex()) + " and " +
              (String)spinBoxMonths.get(spinBoxMonths.getSelectedIndex()));
                super.close();
        }
}
```

Textfelder

Benutzer können ein Textfeld zur Eingabe von Text verwenden.

Art des Textfelds	Beschreibung
email	Zum Einfügen des Zeichens "@" oder eines Punktes (.) in ein E-Mail-Adressenfeld drücken Benutzer die Leertaste.
Datum und Uhrzeit	Bei BlackBerry®-Geräten mit einem Trackwheel oder Trackball können Benutzer das Datum bzw. die Zeit mit der Tastatur oder durch vertikales Drehen des Trackwheel oder des Trackballs ändern. Bei BlackBerry-Geräten mit einem Trackpad können Benutzer das Datum bzw. die Zeit mit der Tastatur oder durch vertikales Bewegen eines Fingers auf dem Trackpad ändern. Bei BlackBerry-Geräten mit einem Touchscreen können Benutzer das Datum bzw. die Zeit durch Anklicken von Datum bzw. Zeit und durch Bewegen eines Fingers nach oben oder unten ändern.
Nummer	Wenn bei BlackBerry-Geräten ohne Touchscreen die Benutzer in ein Nummernfeld eingeben, wechselt das BlackBerry-Gerät in den Modus Zahlensperre, damit zum Eingeben von Zahlen nicht die Alt-Taste gedrückt werden muss. Wenn bei BlackBerry-Geräten mit Touchscreen die Benutzer Nummern in ein Nummernfeld eingeben müssen, wird die Zahlentastatur angezeigt.
Kennwort	Wenn Benutzer in ein Kennwortfeld eingeben, werden anstelle des Textes Sternchen (*) angezeigt. In Kennwortfeldern können Benutzer keinen Text ausschneiden, kopieren oder einfügen, bzw. AutoText nicht verwenden. Bei BlackBerry-Geräten mit SureType®-Technologie ist Mehrfachtippen die standardmäßige Eingabemethode in Kennwortfeldern.

Art des Textfelds	Beschreibung
Telefonnummer	Wenn bei BlackBerry-Geräten ohne Touchscreen die Benutzer in ein Telefonnummernfeld eingeben, wechselt das BlackBerry-Gerät in den Modus Zahlensperre, damit zum Eingeben von Zahlen nicht die Alt-Taste gedrückt werden muss. Sie können es den Benutzern auch erlauben, in Telefonnummernfeldern die folgenden Aktionen auszuführen: • Eingeben des Pluszeichens (+) für internationale Telefonnummern. • Eingeben von Formatierungszeichen wie z. B. das Minuszeichen (-), den Punkt (.), Klammern (()) und Leerzeichen. • Eingeben eines Nummernzeichens (#) oder Sternchens (*). • Zeigen Sie durch Eingeben eines Kommas (,) eine Pause oder eine Wartezeit durch Eingeben eines Ausrufezeichens (!). an. • Zeigen Sie eine Durchwahl durch Drücken der Alt-Taste und Drücken von E, X oder T an. Wenn bei BlackBerry-Geräten mit Touchscreen die Benutzer Nummern in ein Telefonnummernfeld eingeben müssen, wird die Zahlentastatur angezeigt. Sie können es den Benutzern auch erlauben, in Telefonnummernfeldern die folgenden Aktionen auszuführen: • Eingeben eines Nummernzeichens (#) oder Sternchens (*). • Zeigen Sie eine Pause oder eine Durchwahl durch Gedrückthalten der Sternchen-Taste (*) an. • Zeigen Sie eine Wartezeit oder eine Durchwahl durch Gedrückthalten der Nummernzeichen-Taste (#) an.
Text	In Textfeldern geben Benutzer Text ein. Benutzer können in Textfeldern Text ausschneiden, kopieren und einfügen. Wenn der Cursor das Ende einer Textzeile erreicht, bricht der Text in die nächste Zeile um. In Textfeldern können BlackBerry-Geräte Telefonnummern, Webseiten und E-Mail-Adressen auch automatisch in Links umwandeln.
Webadresse	Benutzer können durch Drücken der Leertaste einen Punkt (.) in ein Adressfeld einfügen.

Bewährtes Verfahren: Implementieren von Textfeldern

- Verwenden Sie die `RichTextField`-Klasse, um Textfelder zu erstellen.

- Wählen Sie den Typ des Textfelds basierend darauf aus, welche Art von Eingabe Sie von den Benutzern erwarten. Die Verwendung des geeignetsten Textfelds ist wichtig, damit die entsprechende Eingabeanzeige auf dem Bildschirm angezeigt wird, wenn Benutzer Text eingeben. Wenn Benutzer zum Beispiel Text in ein Feld für Telefonnummern eingeben, wird in der rechten oberen Ecke des Bildschirms die Zahlensperrenanzeige angezeigt. Das Feld, das Sie auswählen, beeinflusst auch die Standardeingabemethode für das Feld auf BlackBerry-Geräten mit SureType®-Technologie.
- Wenn möglich, verwenden oder erweitern Sie vorhandene Felder statt benutzerdefinierte Felder zu erstellen, sodass die Felder das entsprechende Standardverhalten erben.
- Legen Sie einen geeigneten Fokus fest. Wenn möglich, setzen Sie den Fokus auf das Feld, das die Benutzer wahrscheinlich zuerst verwenden werden.
- Verwenden Sie Aufforderungstext nur dann, wenn der Platz auf einem Bildschirm beschränkt ist und Sie keine Bezeichnung und keinen Anweisungstext bereitstellen können. Aufforderungstext wird im Feld angezeigt und verschwindet, wenn die Benutzer Eingaben vornehmen.

Richtlinien für Bezeichnungen

- Verwenden Sie kurze, klare beschreibende Bezeichnungen. Vermeiden Sie Bezeichnungen mit Zeilenumbrüchen.
- Verwenden Sie die übliche Groß-/Kleinschreibung für Titel.
- Schließen Sie die Bezeichnungen von Feldern mit einem Doppelpunkt (:) ab.
- Wenn Sie Aufforderungstext verwenden, formulieren Sie ihn kurz und klar. Verwenden Sie die übliche Groß-/Kleinschreibung für Sätze.

Erstellen eines Textfelds

Erstellen eines schreibgeschützten Textfelds, das eine Formatierung ermöglicht

1. Importieren Sie die `net.rim.device.api.ui.component.RichTextField`-Klasse.
2. Erstellen Sie eine Instanz von `RichTextField`.

```
RichTextField rich = new RichTextField("RichTextField");
```

Erstellen eines bearbeitbaren Textfelds, das keine Formatierung besitzt und Filter akzeptiert

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.ui.component.BasicEditField`
   - `net.rim.device.api.ui.component.EditField`
2. Erstellen Sie eine Instanz von `BasicEditField`.

```
BasicEditField bf = new BasicEditField("BasicEditField: ", "", 10,
EditField.FILTER_UPPERCASE);
```

Erstellen eines bearbeitbaren Textfelds, das Sonderzeichen ermöglicht

1. Importieren Sie die `net.rim.device.api.ui.component.EditField` Klasse.

2. Erstellen Sie eine Instanz von `EditField`.

```
EditField edit = new EditField("EditField: ", "", 10, EditField.FILTER_DEFAULT);
```

Erstellen eines Textfelds für AutoText

Wenn ein Textfeld AutoText unterstützt, wenn Benutzer zweimal die Leertaste drücken, fügt das BlackBerry®-Gerät einen Punkt ein, schreibt den nächsten Buchstaben nach einem Punkt groß und ersetzt Wörter gemäß den Definitionen in der AutoText-Anwendung.

1. Importieren Sie die folgenden Klassen:
   - `net.rim.device.api.ui.component.AutoTextEditField`
   - `net.rim.device.api.ui.autotext.AutoText`
   - `net.rim.device.api.ui.component.BasicEditField`
2. Erstellen Sie eine Instanz von `AutoTextEditField`.

```
AutoTextEditField autoT = new AutoTextEditField("AutoTextEditField: ", "");
```

Erstellen eines Datumsfelds

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import java.lang.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `MyUiScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class MyUi extends UiApplication
{
    public static void main(String[] args)
    {
        MyUi theApp = new MyUi();
        theApp.enterEventDispatcher();
    }
    public MyUi()
    {
        pushScreen(new MyUiScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Bildschirmkonstruktor `setTitle()` auf, um den Titel für den Bildschirm anzugeben.

```
class MyUiScreen extends MainScreen
{
    public MyUiScreen()
    {
        setTitle("UI Component Sample");
    }
}
```

4. Erstellen Sie im Bildschirmkonstruktor mithilfe der `DateField`-Klasse ein Datumsfeld. Stellen Sie `System.currentTimeMillis()` als Parameter bereit, um die aktuelle Uhrzeit zurückzugeben. Verwenden Sie den `DateField.DATE_TIME`-Stil, um sowohl das Datum als auch die Uhrzeit anzuzeigen. Sie können andere Stile verwenden, um nur das Datum oder nur die Uhrzeit anzuzeigen.

```
add(new DateField("Date: ", System.currentTimeMillis(),
    DateField.DATE_TIME));
```

Erstellen eines Kennwortfelds

1. Importieren Sie die `net.rim.device.api.ui.component.PasswordEditField`-Klasse.
2. Erstellen Sie eine Instanz von `PasswordEditField`.
   Die folgende Instanz verwendet zum Beispiel einen Konstruktor, mit dem Sie einen Standardanfangswert für `PasswordEditField` bereitstellen können:

```
PasswordEditField pwd = new PasswordEditField("PasswordEditField: ", "");
```

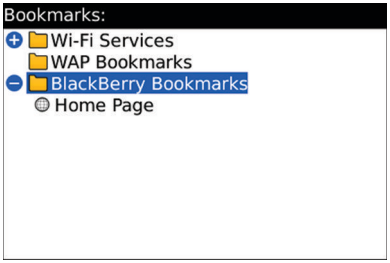
Strukturansichten

Verwenden Sie eine Strukturansicht, um Objekte wie z. B. Ordner, in hierarchischer Weise anzuzeigen.

Objekte in der Strukturansicht sind Knoten. Der oberste Knoten ist der Stammknoten. Ein Knoten in der Struktur kann untergeordnete Knoten besitzen. Ein Knoten, der ein untergeordnetes Element hat, ist ein übergeordneter Knoten.

Benutzer können in einer Strukturansicht die folgende Aktion ausführen:

Aktion	BlackBerry-Geräte ohne Touchscreen	BlackBerry-Geräte mit Touchscreen
Erweitern oder reduzieren Sie ein Objekt mit einem Pluszeichen (+) oder Minuszeichen (-) in einer Hierarchie.	Drücken Sie die Leertaste oder klicken Sie auf das Trackwheel, den Trackball oder das Trackpad.	Klicken Sie auf den Bildschirm.



Bewährtes Verfahren: Implementieren von Strukturansichten

- Verwenden Sie die `TreeField`-Klasse, um Strukturansichten zu erstellen.
- Stellen Sie auf BlackBerry-Geräten mit einem Trackball oder Trackpad ein Kontextmenü bereit, wenn Benutzer mehrere Aktionen ausführen können, wenn sie auf einen übergeordneten Knoten klicken. Wenn Benutzer zum Beispiel auf einen Ordner klicken, der Unterordner und Dateien enthält, sollte ein Kontextmenü angezeigt werden, das Benutzern ermöglicht, entweder den Ordner zu erweitern oder die Dateien anzuzeigen.
- Nehmen Sie nur dann einen Stammknoten auf, wenn die Benutzer Aktionen für die gesamte Struktur ausführen können müssen. Verzichten Sie andernfalls auf den Stammknoten.

Erstellen eines Feldes für das Anzeigen einer Strukturansicht

Verwenden Sie eine Strukturansicht, um Objekte, wie z. B. eine Ordnerstruktur, auf hierarchische Weise anzuzeigen. Ein `TreeField`-Objekt enthält Knoten. Der höchste Knoten ist der Stammknoten. Ein Knoten in der Struktur kann untergeordnete Knoten besitzen. Ein Knoten, der ein untergeordnetes Element besitzt, ist ein übergeordneter Knoten.

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.component.TreeField;
import net.rim.device.api.ui.component.TreeFieldCallback;
import net.rim.device.api.ui.container.MainScreen;
import java.lang.String;
```

2. Implementieren Sie die `TreeFieldCallback`-Schnittstelle.
3. Rufen Sie `TreeField.setExpanded()` für das `TreeField`-Objekt auf, um anzugeben, ob ein Ordner ausblendbar ist. Erstellen Sie ein `TreeField`-Objekt und mehrere untergeordnete Knoten für das `TreeField`-Objekt. Rufen Sie `TreeField.setExpanded()` mit `node4` als Parameter auf, um den Ordner auszublenden.

```
String fieldOne =  new String("Main folder");
...
TreeCallback myCallback = new TreeCallback();
TreeField myTree = new TreeField(myCallback, Field.FOCUSABLE);
int node1 = myTree.addChildNode(0, fieldOne);
int node2 = myTree.addChildNode(0, fieldTwo);
int node3 = myTree.addChildNode(node2, fieldThree);
int node4 = myTree.addChildNode(node3, fieldFour);
...
```

```
int node10 = myTree.addChildNode(node1, fieldTen);
myTree.setExpanded(node4, false);
...
mainScreen.add(myTree);
```

4. Um ein `TreeField`-Objekt neu zu zeichnen, wenn sich ein Knoten ändert, erstellen Sie eine Klasse, die die `TreeFieldCallback`-Schnittstelle implementiert, und implementieren Sie die `TreeFieldCallback.drawTreeItem`-Methode. Die `TreeFieldCallback.drawTreeItem`-Methode verwendet das Cookie für einen Strukturknoten, um an der Position eines Knotens ein `String`-Objekt zu zeichnen. Die `TreeFieldCallback.drawTreeItem`-Methode ruft `Graphics.drawText()` auf, um das `String`-Objekt zu zeichnen.

```
private class TreeCallback implements TreeFieldCallback
{
    public void drawTreeItem(TreeField _tree, Graphics g, int node, int y, int
width, int indent)
    {
        String text = (String)_tree.getCookie(node);
        g.drawText(text, indent, y);
    }
}
```

Menübefehle

8

Erstellen eines Menüs

Die `MainScreen`-Klasse stellt Standardkomponenten einer BlackBerry®-Geräteanwendung bereit. Sie enthält ein Standardmenü.

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `CreateMenuScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class CreateMenu extends UiApplication
{
   public static void main(String[] args)
   {
      CreateMenu theApp = new CreateMenu();
      theApp.enterEventDispatcher();
   }
   public CreateMenu()
   {
      pushScreen(new CreateMenuScreen());
   }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Bildschirmkonstruktor `setTitle()` auf, um den Titel für den Bildschirm anzugeben. Rufen Sie `add()` auf, um dem Bildschirm ein Textfeld hinzuzufügen. Rufen Sie `addMenuItem()` auf, um dem von `MainScreen` erstellten Menü einen Menüeintrag hinzuzufügen.

```
class CreateMenuScreen extends MainScreen
{
   public CreateMenuScreen()
   {
      setTitle("Create Menu Sample");
      add(new RichTextField("Create a menu"));
      addMenuItem(_viewItem);
   }
}
```

4. Erstellen Sie den Menüeintrag mithilfe der `MenuItem`-Klasse. Setzen Sie `run()` außer Kraft, um die Aktion anzugeben, die ausgeführt wird, wenn der Benutzer auf den Menüeintrag klickt. Wenn der Benutzer auf den Menüeintrag klickt, ruft die Anwendung `Menu.run()` auf.

```
private MenuItem _viewItem = new MenuItem("More Info", 110, 10)
{
   public void run()
   {
      Dialog.inform("Display more information");
   }
};
```

5. Setzen Sie `Menu()` außer Kraft, um ein Dialogfeld anzuzeigen, wenn der Benutzer auf den Menüeintrag "Schließen" klickt. Standardmäßig ist der Menüeintrag "Schließen" in dem von `MainScreen` erstellten Menü enthalten. Rufen Sie `super.close()` auf, um die Anwendung zu schließen. Wenn der Benutzer das Dialogfeld schließt, ruft die Anwendung `MainScreen.close()` auf, um die Anwendung zu schließen.

```
public void close()
{
    Dialog.alert("Goodbye!");
    super.close();
}
```

Codebeispiel: Erstellen eines Menüs

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
public class CreateMenu extends UiApplication
{
     public static void main(String[] args)
     {
         CreateMenu theApp = new CreateMenu();
         theApp.enterEventDispatcher();
     }
     public CreateMenu()
     {
         pushScreen(new CreateMenuScreen());
     }
}
class CreateMenuScreen extends MainScreen
{
    public CreateMenuScreen()
    {
      setTitle("Create Menu Sample");
      add(new RichTextField("Create a menu"));
      addMenuItem(_viewItem);
    }
    private MenuItem _viewItem = new MenuItem("More Info", 110, 10)
```

```
    {
        public void run()
        {
            Dialog.inform("Display more information");
        }
    };
    public void close()
    {
        Dialog.alert("Goodbye!");
        super.close();
    }
}
```

Bewährtes Verfahren: Implementieren von Menüs

- Stellen Sie stets ein vollständiges Menü bereit.
- Stellen Sie sicher, dass Benutzer die Menütaste drücken können, um das vollständige Menü zu öffnen, und eine Aktion zu initiieren, wenn ein Menüelement markiert ist. Stellen Sie sicher, dass Benutzer auch die Menütaste halten können, um das Dialogfeld zum Wechseln von Anwendungen zu öffnen.
- Verwenden Sie als Standardmenüelement das Menüelement, das die Benutzer am wahrscheinlichsten auswählen werden.
- Platzieren Sie den Standardmenüeintrag und die anderen am häufigsten verwendeten Menüeinträge in der Mitte des Menüs.
- Vergewissern Sie sich, dass die Reihenfolge der Menüeinträge der Reihenfolge der Menüeinträge in anderen BlackBerry®-Geräteanwendungen entspricht.
- Gruppieren Sie Menüeinträge entsprechend der Verwendungsweise oder der Funktionalität und lassen Sie Ihre Gruppierungen wenn möglich von den Benutzern testen.
- Fügen Sie Trennzeichen zwischen den Menüeintragsgruppen ein.
- Platzieren Sie keine Menüeinträge für einander widersprechende Aktionen nahe beieinander. Platzieren Sie zum Beispiel nicht den Menüeintrag "Löschen" neben dem Menüeintrag "Öffnen".
- Stellen Sie stets die Menüeinträge "Anwendung wechseln" und "Schließen" bereit. Platzieren Sie diese Menüeinträge am Ende des Menüs. Wenn Sie Standardkomponenten verwenden, sind diese Menüeinträge automatisch enthalten.

Richtlinien für Bezeichnungen

- Verwenden Sie kurze und klare beschreibende Bezeichnungen, die nicht länger als 12 Zeichen sind. Wenn eine Bezeichnung zu lang ist, werden Auslassungspunkte (...) angezeigt, um zu verdeutlichen, dass der Text abgeschnitten ist.
- Verwenden Sie Verben für Bezeichnungen.
- Verwenden Sie die übliche Groß-/Kleinschreibung für Bezeichnungen.
- Verwenden Sie Auslassungspunkte in einer Menüeintragsbezeichnung, um anzuzeigen, dass Benutzer nach dem Klicken auf den Menüeintrag eine weitere Aktion ausführen müssen. Wenn Benutzer zum Beispiel im Kalender auf den Menüeintrag "Wechseln zu Datum..." klicken, müssen sie auf dem eingeblendeten Bildschirm ein Datum angeben.
- Vermeiden Sie in Bezeichnungen die Verwendung von Symbolen wie z. B. Sternchen (*).

Hinzufügen eines Menüeintrags zu einer BlackBerry Device Software-Anwendung

Sie können einer BlackBerry® Device Software-Anwendung mithilfe der API für Menüeinträge im `net.rim.blackberry.api.menuitem`-Paket einen Menüeintrag hinzufügen. Sie können zum Beispiel der Kontaktanwendung auf einem BlackBerry-Gerät einen Menüeintrag namens "Auftrag anzeigen" hinzufügen, damit, wenn ein Benutzer auf den Menüeintrag klickt, eine CRM-Anwendung geöffnet und eine Liste der Aufträge für diesen Kontakt angezeigt wird.

Die `ApplicationMenuItemRepository`-Klasse stellt die Konstanten bereit, die die BlackBerry Device Software-Anwendung angeben, in der Ihr Menüeintrag angezeigt werden sollte. Die `MENUITEM_MESSAGE_LIST`-Konstante gibt zum Beispiel an, dass der Menüeintrag in der Nachrichtenanwendung angezeigt werden soll.

Hinzufügen eines Menüeintrags zu einer BlackBerry Device Software-Anwendung

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.blackberry.api.menuitem.*;
import net.rim.blackberry.api.pdap.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
```

2. Erweitern Sie die abstrakte `ApplicationMenuItem`-Klasse, um einen Menüeintrag zu erstellen. Setzen Sie den `ApplicationMenuItem()`-Konstruktor mit einer Ganzzahl außer Kraft, um die Position des Menüeintrags im Menü anzugeben. Durch eine höhere Zahl wird der Menüeintrag weiter unten im Menü platziert.

```
public class SampleMenuItem extends ApplicationMenuItem
{
   SampleMenuItem()
   {
      super(20);
   }
}
```

3. Implementieren Sie `toString()`, um den Menüeintragstext anzugeben.

```
public String toString()
{
   return "Open the Contacts Demo application";
}
```

4. Rufen Sie `getInstance()` auf, um die Anwendungsdatenbank abzurufen.

```
ApplicationMenuItemRepository repository =
    ApplicationMenuItemRepository.getInstance();
```

5. Erstellen Sie eine Instanz einer Klasse, um die `MenuItem`-Klasse zu erweitern.

```
ContactsDemoMenuItem contactsDemoMenuItem = new
    ContactsDemoMenuItem();
```

6. Rufen Sie `ApplicationMenuItemRepository.addMenuItem()` auf, um den Menüeintrag der entsprechenden BlackBerry®-Geräteanwendungs-Datenbank hinzuzufügen.

```
repository.addMenuItem(ApplicationMenuItemRepository
    MENUITEM_ADDRESSCARD_VIEW, contactsDemoMenuItem);
```

7. Implementieren Sie `run()`, um das Verhalten des Menüeintrags festzulegen. Wenn im folgenden Codebeispiel ein Benutzer auf den neuen Menüeintrag klickt und ein `Contact`-Objekt vorhanden ist, empfängt die `ContactsDemo`-Anwendung das Ereignis und ruft `ContactsDemo.enterEventDispatcher()` auf.

```
public Object run(Object context)
{
   BlackBerryContact c = (BlackBerryContact)context;
   if ( c != null )
   {
      new ContactsDemo().enterEventDispatcher();
   }
   else
   {
      throw new IllegalStateException( "Context is null, expected a Contact
     instance");
   }
   Dialog.alert("Viewing an email message in the email view");
   return null;
}
```

Ändern der Darstellung eines Menüs

Sie können den Hintergrund, Rahmen und die Schriftart von einem Menü mithilfe der `Menu`-Klasse im `net.rim.device.api.ui.component`-Paket ändern. Zum Beispiel können Sie die Anzeige vom Menü ändern, damit es ein Erscheinungsbild hat, das ähnlich zum Rest von Ihrer BlackBerry® -Geräteanwendung ist. Wenn Sie die Anzeige von einem Menü ändern, hebt Ihre BlackBerry-Geräteanwendung das Thema auf, das auf dem BlackBerry-Gerät festgelegt ist.

Ändern der Darstellung eines Menüs

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.decor.*;
import net.rim.device.api.system.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `CreateCustomMenuScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class CreateCustomMenu extends UiApplication
{
   public static void main(String[] args)
   {
      CreateCustomMenu theApp = new CreateCustomMenu();
      theApp.enterEventDispatcher();
   }
   public CreateCustomMenu()
   {
      pushScreen(new CreateCustomMenuScreen());
   }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Bildschirmkonstruktor `setTitle()` auf, um den Titel für den Bildschirm anzugeben. Rufen Sie `add()` auf, um auf dem Bildschirm ein Textfeld hinzuzufügen.

```
class CreateCustomMenuScreen extends MainScreen
{
    Background _menuBackground;
    Border _menuBorder;
    Font _menuFont;
    CreateCustomMenuScreen()
    {
        setTitle("Custom Menu Sample");
        add(new RichTextField("Creating a custom menu"));
    }
}
```

4. Geben Sie im Bildschirmkonstruktor die Darstellung des Menüs an. Legen Sie durch Erstellen eines `XYEdges`-Objekt die Abstände für den Rahmen um das Menü fest. Rufen Sie `createRoundedBorder()` auf, um einen Rahmen mit abgerundeten Ecken zu erstellen. Rufen Sie `createSolidTransparentBackground()` auf, um eine transparente Hintergrundfarbe für das Menü zu erstellen.

```
XYEdges thickPadding = new XYEdges(10, 10, 10, 10);
_menuBorder = BorderFactory.createRoundedBorder(thickPadding,
Border.STYLE_DOTTED);
_menuBackground = BackgroundFactory.createSolidTransparentBackground(Color
    LIGHTSTEELBLUE, 50);
```

5. Geben Sie im Bildschirmkonstruktor die Schriftart für das Menü mithilfe eines `FontFamily`-Objekts an. Rufen Sie `forName()` auf, um eine Schriftart vom BlackBerry®-Gerät abzurufen. Rufen Sie `getFont()` auf, um den Schnitt und die Größe der Schriftart anzugeben.

```
try
{
    FontFamily family = FontFamily.forName("BBCasual");
    _menuFont = family.getFont(Font.PLAIN, 30, Ui.UNITS_px);
}
catch(final ClassNotFoundException cnfe)
{
    UiApplication.getUiApplication().invokeLater(new Runnable()
    {
        public void run()
        {
            Dialog.alert("FontFamily.forName() threw " + cnfe.toString());
        }
    });
}
```

6. Setzen Sie in der Bildschirmklasse die `makeMenu()`-Methode außer Kraft, um die Darstellung des Menüs anzuwenden. Rufen Sie `setBackground()`, `setBorder()` und `setFont()` auf, um die Darstellung des Menüs anzuwenden, die Sie in den Schritten 4 und 5 angegeben haben.

```
protected void makeMenu(Menu menu, int context)
{
    menu.setBorder(_menuBorder);
    menu.setBackground(_menuBackground);
    menu.setFont(_menuFont);
    super.makeMenu(menu, context);
}
```

Codebeispiel: Ändern der Darstellung eines Menüs

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.decor.*;
import net.rim.device.api.system.*;
public class CreateCustomMenu extends UiApplication
{
   public static void main(String[] args)
   {
      CreateCustomMenu theApp = new CreateCustomMenu();
      theApp.enterEventDispatcher();
   }
   public CreateCustomMenu()
   {
      pushScreen(new CreateCustomMenuScreen());
   }
}
class CreateCustomMenuScreen extends MainScreen
{
    Border _menuBorder;
```

```
    Background _menuBackground;
    Font _menuFont;
    CreateCustomMenuScreen()
    {
        setTitle("Custom Menu Sample");
        add(new RichTextField("Creating a custom menu"));
        XYEdges thickPadding = new XYEdges(10, 10, 10, 10);
        _menuBorder = BorderFactory.createRoundedBorder(thickPadding,
    Border.STYLE_DOTTED);
        _menuBackground = BackgroundFactory.createSolidTransparentBackground(Color
    LIGHTSTEELBLUE, 50);
        try
        {
            FontFamily family = FontFamily.forName("BBCasual");
            _menuFont = family.getFont(Font.PLAIN, 30, Ui.UNITS_px);
        }
        catch(final ClassNotFoundException cnfe)
        {
            UiApplication.getUiApplication().invokeLater(new Runnable()
            {
                public void run()
                {
                    Dialog.alert("FontFamily.forName() threw " + cnfe.toString());
                }
            });
        }
    }
    protected void makeMenu(Menu menu, int context)
    {
        menu.setBorder(_menuBorder);
        menu.setBackground(_menuBackground);
        menu.setFont(_menuFont);
        super.makeMenu(menu, context);
    }
}
```

Hinzufügen eines Symbols zu einem Menüeintrag

Sie können einem Menüelement ein Symbol mithilfe des `MenuItem`s im `Net.rim.device.api.ui`-Pakets hinzufügen. Das BlackBerry® -Gerät berechnet die Größe vom Symbol und zentriert das Symbol nach links oder rechts vom Menüelementtext, wenn es das Menü anlegt. Das Symbol richtet sich nach links oder rechts vom Menüelementtext abhängig von der Textrichtung aus (zum Beispiel bei der Textrichtung von links nach rechts, richtet sich das Symbol links vom Text aus). Falls notwendig ändert das BlackBerry-Gerät die Höhe und Breite des Symbols in Beziehung zur Höhe der Standardschriftart, die die BlackBerry-Geräteanwendung verwendet. Wenn das BlackBerry-Gerät die Größe des Bilds ändert, bewahrt es das Bildformat vom Bild.

Anmerkung: Sie können einem vordefinierten Menüelement kein Symbol hinzufügen (zum Beispiel `MenuItem.SAVE_CLOSE`).

Hinzufügen eines Symbols zu einem Menüeintrag

Vor Beginn erforderliche Aktion: Fügen Sie das Symbol Ihrem Projekt hinzu.

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.image.*;
import net.rim.device.api.system.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `AddIconScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class AddIcon extends UiApplication
{
   public static void main(String[] args)
   {
      AddIcon theApp = new AddIcon();
      theApp.enterEventDispatcher();
   }
   public AddIcon()
   {
      pushScreen(new AddIconScreen());
   }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Bildschirmkonstruktor `setTitle()` auf, um den Titel für den Bildschirm anzugeben. Rufen Sie `add()` auf, um dem Bildschirm ein Textfeld hinzuzufügen. Rufen Sie `addMenuItem()` auf, um dem von `MainScreen` erstellten Menü einen Menüeintrag hinzuzufügen.

```
class AddIconScreen extends MainScreen
{
    public AddIconScreen()
    {
      setTitle("Menu Icon Sample");
      add(new RichTextField("Add an icon to a menu"));
      addMenuItem(_viewItem);
    }
}
```

4. Geben Sie im Bildschirmkonstruktor mithilfe der `Image`-Klasse das Symbol an, das Sie dem Menüeintrag hinzufügen möchten. Rufen Sie `createImage()` auf, um das Symbol für den Menüeintrag zu erstellen. Rufen Sie `setIcon()` auf, um das Symbol dem Menüeintrag hinzuzufügen.

```
Image menuIcon =
    ImageFactory.createImage(Bitmap.getBitmapResource("View.jpg"));
_viewItem.setIcon(menuIcon);
```

5. Erstellen Sie in der Bildschirmklasse mithilfe der `MenuItem`-Klasse den Menüeintrag. Setzen Sie `run()` außer Kraft, um die Aktion anzugeben, die ausgeführt wird, wenn der Benutzer auf den Menüeintrag klickt.

```
private MenuItem _viewItem = new MenuItem("More Info", 110, 10)
{
   public void run()
   {
      Dialog.inform("Display more information");
   }
};
```

Codebeispiel: Hinzufügen eines Symbols zu einem Menüelement

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.image.*;
import net.rim.device.api.system.*;
public class AddIcon extends UiApplication
{
    public static void main(String[] args)
    {
        AddIcon theApp = new AddIcon();
        theApp.enterEventDispatcher();
    }
    public AddIcon()
    {
        pushScreen(new AddIconScreen());
    }
}
class AddIconScreen extends MainScreen
{
    public AddIconScreen()
    {
     setTitle("Menu Icon Sample");
     add(new RichTextField("Add an icon to a menu"));
     Image menuIcon = ImageFactory.createImage(Bitmap.getBitmapResource("View.jpg"));
     _viewItem.setIcon(menuIcon);
     addMenuItem(_viewItem);
  }
  private MenuItem _viewItem = new MenuItem("More Info", 110, 10)
  {
         public void run()
         {
              Dialog.inform("Display more information");
```

```
            }
    };
}
```

Benutzerdefinierte Schriftarten

9

Die `FontManager`-Klasse im `net.rim.device.api.ui`-Paket stellt Konstanten und Methoden bereit, die Sie verwenden können, um eine TrueType-Schriftartauf einem BlackBerry®-Gerät zu installieren und zu deinstallieren. Die maximale Größe, die für die TrueType-Schriftartdatei ermöglicht wird, ist 60 KB. Sie können angeben, ob die Schriftart für die Anwendung, die die Schriftart installiert oder für alle Anwendungen auf dem BlackBerry-Gerät verfügbar ist.

Die `FontManager`-Klasse stellt auch Methoden bereit, um die Standardschriftart für das BlackBerry-Gerät oder die Anwendung festzulegen.

Installieren und Verwenden einer benutzerdefinierten Schriftart in einer BlackBerry-Java-Anwendung

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.component.*;
import java.util.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `FontLoadingDemoScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class FontLoadingDemo extends UiApplication
{
    public static void main(String[] args)
    {
        FontLoadingDemo app = new FontLoadingDemo();
        app.enterEventDispatcher();
    }
    public FontLoadingDemo()
    {
        pushScreen(new FontLoadingDemoScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm durch Erweitern der `MainScreen`-Klasse. Rufen Sie `setTitle()` auf, um den Text festzulegen, der im Titelbereich des Bildschirms angezeigt wird. Erstellen Sie ein neues `LabelField`-Objekt. Sie wenden die benutzerdefinierte Schriftart auf dieses Objekt an.

```
class FontLoadingDemoScreen extends MainScreen
{
    public FontLoadingDemoScreen()
    {
        setTitle("Font Loading Demo");
        LabelField helloWorld = new LabelField("Hello World");
    }
}
```

4. Rufen Sie im Bildschirmkonstruktor die `FontManager.getInstance()`-Methode auf, um einen Verweis auf das `FontManager`-Objekt zu erhalten, und rufen Sie anschließend die `load()`-Methode auf, um die Schriftart zu installieren. Schließen Sie den `load()`-Aufruf in eine IF-Anweisung ein, um zu prüfen, ob die Installation erfolgreich war. Die `load()`-Methode gibt eine Markierung zurück, die angibt, ob die Schriftart erfolgreich installiert wurde. Der folgende Code gibt an, dass die Schriftart nur von der Anwendung verwendet werden kann.

```
if (FontManager.getInstance().load("Myfont.ttf", "MyFont",
    FontManager.APPLICATION_FONT) == FontManager.SUCCESS)
{
}
```

5. Erstellen Sie im Bildschirmkonstruktor in einem try/catch-Block in der in Schritt 5 erstellten IF-Anweisung ein `Font`-Objekt für die Schriftart, die Sie gerade installiert haben. Rufen Sie die `setFont()`-Methode auf, um die in Schritt 5 erstellte Schriftart auf das `LabelField`-Objekt anzuwenden.

```
try
{
    FontFamily family = FontFamily.forName("MyFont");
    Font myFont = family.getFont(Font.PLAIN, 50);
    helloWorld.setFont(myFont);
}
catch (ClassNotFoundException e)
{
    System.out.println(e.getMessage());
}
```

6. Rufen Sie im Bildschirmkonstruktor `add()` auf, um das `LabelField`-Objekt dem Bildschirm hinzuzufügen.

```
add(helloWorld);
```

Codebeispiel: Installieren und Verwenden einer benutzerdefinierten Schriftart in einer BlackBerry-Java-Anwendung

```
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.component.*;
import java.util.*;
public class FontLoadingDemo extends UiApplication
```

```
{
    public static void main(String[] args)
    {
        FontLoadingDemo app = new FontLoadingDemo();
        app.enterEventDispatcher();
    }
    public FontLoadingDemo()
    {
        pushScreen(new FontLoadingDemoScreen());
    }
}
class FontLoadingDemoScreen extends MainScreen
{
    public FontLoadingDemoScreen()
    {
        setTitle("Font Loading Demo");
        LabelField helloWorld = new LabelField("Hello World");
        if (FontManager.getInstance().load("Myfont.ttf", "MyFont",
    FontManager.APPLICATION_FONT) == FontManager.SUCCESS)
        {
            try
            {
                FontFamily typeface = FontFamily.forName("Charming");
                Font myFont = typeface.getFont(Font.PLAIN, 50);
                helloWorld.setFont(myFont);
            }
            catch (ClassNotFoundException e)
            {
                System.out.println(e.getMessage());
            }
        }
        add(helloWorld);
    }
}
```

Rechtschreibprüfung

10

Sie können die Elemente im Paket `net.rim.blackberry.api.spellcheck` verwenden, um einer Anwendung die Funktion zur Rechtschreibprüfung hinzuzufügen. Die `SpellCheckEngine`-Schnittstelle befähigt eine Anwendung, die Rechtschreibung eines UI-Feld-Wertes zu prüfen, und gibt einem BlackBerry®-Gerätebenutzer die Optionen zur Rechtschreibkorrektur. Die `SpellCheckUI`-Schnittstelle befähigt eine Anwendung, eine Benutzeroberfläche bereitzustellen, die es einem BlackBerry-Gerätebenutzer ermöglicht, ein Rechtschreibeproblem durch Interagieren mit der `SpellCheckEngine`-Implementierung zu lösen.

Weitere Informationen zur Verwendung der Spell-Check-API können Sie der Spell-Check-Beispielanwendung entnehmen, die im BlackBerry® Java® Development Environment 4.3.1 oder neuer sowie im BlackBerry® Java® Plug-in für Eclipse® enthalten ist.

Hinzufügen einer Funktion zur Rechtschreibprüfung

1. Importieren Sie die folgenden Klassen:
   - `net.rim.blackberry.api.spellcheck.SpellCheckEngineFactory`
   - `java.lang.StringBuffer`
2. Importieren Sie die folgenden Schnittstellen:
   - `net.rim.blackberry.api.spellcheck.SpellCheckEngine`
   - `net.rim.blackberry.api.spellcheck.SpellCheckUI`
   - `net.rim.blackberry.api.spellcheck.SpellCheckUIListener`
3. Erstellen Sie Variablen für Rechtschreibprüfungsobjekte.

```
SpellCheckEngine _spellCheckEngine;
SpellCheckUI _spellCheckUI;
```

4. Rufen Sie `createSpellCheckUI()` auf.

```
_spellCheckUI = SpellCheckEngineFactory.createSpellCheckUI();
```

5. Um eine Anwendung zu benachrichtigen, wenn ein Rechtschreibprüfungsereignis auftritt, rufen Sie `addSpellCheckUIListener()` mit einem `SpellCheckUIListener`-Objekt als Parameter auf.

```
_spellCheckUI.addSpellCheckUIListener(new SpellCheckUIListener());
```

6. Um eine Anwendung eine Rechtschreibprüfung für Benutzeroberflächenfelder durchführen und BlackBerry-Gerätebenutzern Rechtschreibkorrekturen vorschlagen zu lassen, rufen Sie ein `SpellCheckEngine`-Objekt ab und rufen Sie `getSpellCheckEngine()` auf.

```
_spellCheckEngine = _spellCheckUI.getSpellCheckEngine();
```

7. Um eine Korrektur für ein falsch buchstabiertes Wort zu verwenden, rufen Sie `SpellCheckEngine.learnCorrection()` auf. Verwenden Sie die Parameter `new StringBuffer(Text)` und `new StringBuffer(Korrektur)`, wobei `Text` für das falsch buchstabierte Wort und `Korrektur` für das richtig buchstabierte Wort steht.

```
_spellCheckEngine.learnCorrection(new StringBuffer(text), new StringBuffer
(correction));
```

8. Um für ein Feld Rechtschreibprüfungsvorgänge auszuführen, rufen Sie `SpellCheckUI.spellCheck()` mit einem `field`-Objekt als Parameter auf.

```
_spellCheckUI.spellCheck(field);
```

9. Um ein falsch buchstabiertes Wort als richtig buchstabiertes Wort zu akzeptieren, rufen Sie `SpellCheckEngine.learnWord()` mit dem zu lernenden Wort als Parameter auf.

```
_spellCheckEngine.learnWord(new StringBuffer(word));
```

Empfangen von Rechtschreibprüfungsereignissen

1. Importieren Sie die folgenden Klassen:
   - `java.lang.StringBuffer`
   - `net.rim.device.api.ui.UiApplication`
   - `net.rim.device.api.ui.Field`
2. Importieren Sie die folgenden Schnittstellen:
   - `net.rim.blackberry.api.spellcheck.SpellCheckUIListener`
   - `net.rim.blackberry.api.spellcheck.SpellCheckEngine`
3. Erstellen Sie eine Methode, die die `SpellCheckUIListener.LEARNING_ACCEPT`-Konstante zurückgibt, wenn `SpellCheckEngine` ein neues Wort lernt.

```
public int wordLearned(SpellCheckUI ui, StringBuffer word) {
   UiApplication.getUiApplication().invokeLater(new popUpRunner("Word learned"));
   return SpellCheckUIListener.LEARNING_ACCEPT;
}
```

4. Erstellen Sie eine Methode, die die `SpellCheckUIListener.LEARNING_ACCEPT`-Konstante zurückgibt, wenn `SpellCheckEngine` eine Korrektur für ein Wort lernt.

```
public int wordCorrectionLearned(SpellCheckUI ui, StringBuffer word, StringBuffer
correction){
   UiApplication.getUiApplication().invokeLater(new popUpRunner("Correction
learned"));
   return SpellCheckUIListener.LEARNING_ACCEPT;
}
```

5. Erstellen Sie eine Methode, die die `SpellCheckUIListener.ACTION_OPEN_UI`-Konstante zurückgibt, wenn `SpellCheckEngine` ein falsch buchstabiertes Wort findet.

```
public int misspelledWordFound(SpellCheckUI ui, Field field, int offset, int len){
   UiApplication.getUiApplication().invokeLater(new popUpRunner("Misspelled word
found"));
   return SpellCheckUIListener.ACTION_OPEN_UI;
}
```

Verwandte Ressourcen

11

Weitere Informationen zu Benutzeroberfläche und Navigationsentwicklung für BlackBerry®-Geräte unter www.blackberry.com/go/docs/developers Weitere Informationen finden Sie hier:

- API-Referenz für BlackBerry® Java® Development Environment
- Basishandbuch für BlackBerry Java Application
- Entwicklungshandbuch für BlackBerry Java Application
- BlackBerry Java Application Accessibility Development Guide (Entwicklungshandbuch für BlackBerry Java Application – Barrierefreiheit)
- BlackBerry Java Application Integration Development Guide (Entwicklungshandbuch für BlackBerry Java Application – Integration)
- BlackBerry Java Application Multimedia Development Guide (Multimedia-Entwicklungshandbuch für BlackBerry Java Application)
- BlackBerry Java Development Environment Transitioning to Touch Screen Development Technical Note (Technischer Hinweis für BlackBerry Java Development Environment – Übergang zur Touchscreen-Entwicklung)
- BlackBerry Java Development Environment Überblick über Beispielanwendungen
- BlackBerry Developer Zone unter www.blackberry.com/developers
- BlackBerry-Entwickler können Sie herunterladen unter www.blackberry.com/developers/downloads

Glossar

12

3D
Dreidimensional

API
Application Programming Interface (wendungsprogrammierschnittstelle)

JVM
Java® Virtual Machine

MIDP
Mobile Information Device Profile (Softwareprofil für mobile Geräte)

Rückmeldung

13

Falls Sie uns eine Rückmeldung zu diesem Artikel zukommen lassen möchten, besuchen Sie die Website www.blackberry.com/docsfeedback.

Verlauf der Dokumentrevision

14

Datum	Beschreibung
6. April 2010	Die folgenden Themen wurden hinzugefügt: • Anordnen von Benutzeroberflächenkomponenten • Autocomplete-Textfeld • Schaltflächen • Kontrollkästchen • Dialogfelder • Dropdown-Listen • Interaktionsmethoden auf BlackBerry-Geräten • Tastatur • Bezeichnungen • Listen • Optionsfelder • Auswahldialogfelder • Fortschrittsanzeigen • Suchfelder • Drehungsfelder • Textfelder • Trackball oder Trackpad • Strukturansichten Die folgenden Codebeispiele wurden hinzugefügt: • Codebeispiel: Konfigurieren eines Browserfelds • Codebeispiel: Erstellen eines Kontrollkästchens • Codebeispiel: Erstellen einer Dropdown-Liste • Codebeispiel: Erstellen eines Optionsfelds • Codebeispiel: Anzeigen des HTML-Inhalts von einer Ressource in Ihrer Anwendung • Codebeispiel: Anzeigen von HTML-Inhalt in einem Browserfeld

Datum	Beschreibung
	• Codebeispiel: Anzeigen von HTML-Inhalt von einer Webseite in einem Browserfeld • Codebeispiel: Senden von Formulardaten an eine Webadresse in einem Browserfeld Die folgenden Themen wurden geändert: • Erstellen eines Kontrollkästchens • Erstellen eines Datumsfelds • Erstellen einer Dropdown-Liste • Erstellen einer Optionstaste • Reagieren auf einen Benutzer, der den Bildschirm zweimal schnell berührt • Festlegen des Formats und der Richtung des Bildschirms • Rechtschreibprüfung Die folgenden Themen wurden gestrichen: • Bewährtes Verfahren: Implementieren von Untermenüs • Codebeispiel: Erstellen eines Untermenüs • Codebeispiel: Anzeigen einer Bezeichnung an einer absoluten Position auf dem Bildschirm • Erstellen eines Layout-Managers • Erstellen eines Bildschirms • Erstellen eines Untermenüs • Anzeigen eines Felds an einer absoluten Position auf einem Bildschirm • Anzeigen einer Bezeichnung an einer absoluten Position auf dem Bildschirm • Bereitstellen von Bildschirmnavigation beim Verwenden eines Mainscreen
9. November 2009	Die folgenden Themen wurden hinzugefügt: • Erstellen eines Raster-Layouts • Erstellen einer Gittergröße • Erstellen einer Dateiauswahl • Verwenden von Datenquellen und Feldern mit einem autocomplete-Textfeld Die folgenden Codebeispiele wurden hinzugefügt:

Datum	Beschreibung
	• Codebeispiel: Erstellen eines Raster-Layouts • Codebeispiel: Erstellen eines Dateipickers
6. Oktober 2009	Die folgenden Themen wurden hinzugefügt: • Bewährtes Verfahren: Implementieren von Untermenüs • Erstellen eines autocomplete-Textfelds aus einem Datensatz • Erstellen eines autocomplete-Textfelds aus einer Datenquelle • Erstellen einer Datumsauswahl • Erstellen eines Drehfelds • Erstellen eines Bildschirmübergangs • Benutzerdefinierte Schriftarten • Installieren und Verwenden einer benutzerdefinierten Schriftart in einer BlackBerry-Java-Anwendung Die folgenden Codebeispiele wurden hinzugefügt: • Codebeispiel: Erstellen eines autocomplete-Felds aus einem Datensatz • Codebeispiel: Erstellen eines autocomplete-Felds aus einem Datensatz • Codebeispiel: Erstellen eines Bildschirmübergangs • Codebeispiel: Erstellen eines Drehfelds • Codebeispiel: Installieren und Verwenden einer benutzerdefinierten Schriftart in einer BlackBerry-Java-Anwendung
14. August 2009	Die folgenden Themen wurden hinzugefügt: • Hinzufügen eines Symbols zu einem Menüeintrag • Hinzufügen eines Symbols zu einem Menüeintrag • Ändern der Darstellung eines Menüs • Ändern der Darstellung eines Menüs • Konfigurieren eines Browserfeldes • Anzeigen von HTML-Inhalt in einem Browserfeld • Erstellen eines Menüs • Erstellen eines Untermenüs • Erstellen eines Felds, um Webinhalte anzuzeigen • Anzeigen eines Felds an einer absoluten Position auf einem Bildschirm

Datum	Beschreibung
	• Vorübergehendes Anzeigen eines ButtonField und eines LabelField oben bzw. unten am Bildschirm • Anzeigen von HTML-Inhalt aus einer Ressource in Ihrer Anwendung • Anzeigen von HTML-Inhalt von einer Webseite in einem Browserfeld • Anzeigen einer Bezeichnung an einer absoluten Position auf dem Bildschirm • Anzeigen von Feldern in einem temporären Paar von Managern • Ereignisinjektion • Senden von Formulardaten an eine Webadresse in einem Browserfeld Die folgenden Codebeispiele wurden hinzugefügt: • Codebeispiel: Hinzufügen eines Symbols zu einem Menüelement • Codebeispiel: Ändern der Darstellung eines Menüs • Codebeispiel: Erstellen einer Datumsauswahl • Codebeispiel: Erstellen eines Menüs • Codebeispiel: Erstellen eines Untermenüs • Codebeispiel: Vorübergehendes Anzeigen eines ButtonField und eines LabelField oben bzw. unten am Bildschirm • Codebeispiel: Anzeigen einer Bezeichnung an einer absoluten Position auf dem Bildschirm Die folgenden Themen wurden geändert: • Hinzufügen eines Menüeintrags zu einer BlackBerry Device Software-Anwendung • Hinzufügen eines Menüeintrags zu einer BlackBerry Device Software-Anwendung • Benachrichtigen lassen, wenn sich die Größe des Zeichenbereichs auf dem Touchscreen ändert • Abrufen von Beschleunigungssensordaten • Abrufen der Beschleunigungssensordaten in bestimmten Abständen • Festlegen des Formats und der Richtung des Bildschirms • Speichern der ausgelesenen Beschleunigungssensorwerte in einem Puffer

Rechtliche Hinweise

15

©2010 Research In Motion Limited. Alle Rechte vorbehalten. BlackBerry®, RIM®, Research In Motion®, SureType®, SurePress™ und die zugehörigen Marken, Namen und Logos sind Eigentum von Research In Motion Limited und sind in den USA und weiteren Ländern weltweit als Marken eingetragen und/oder werden dort als Marken verwendet.

Eclipse ist eine Marke der Eclipse Foundation, Inc. Java ist eine Marke von Sun Microsystems, Inc. TrueType ist eine Marke von Apple Inc. Alle weiteren Marken sind Eigentum ihrer jeweiligen Inhaber.

Dieses Dokument und alle Dokumente, die per Verweis in dieses Dokument mit einbezogen werden, z. B. alle unter www.blackberry.com/go/docs erhältlichen Dokumente werden ohne Mängelgewähr und je nach Verfügbarkeit bereitgestellt. Die entsprechenden Dokumente werden ohne ausdrückliche Billigung, Gewährleistung oder Garantie seitens Research In Motion Limited und seinen angegliederten Unternehmen ("RIM") bereitgestellt. RIM übernimmt keine Verantwortung für eventuelle typografische, technische oder anderweitige Ungenauigkeiten sowie für Fehler und Auslassungen in den genannten Dokumenten. Dieses Dokument beschreibt eventuell einige Aspekte der RIM-Technologie in allgemeiner Form, um das Eigentum und die vertraulichen Informationen und/oder Geschäftsgeheimnisse von RIM zu schützen. RIM behält sich das Recht vor, die in diesem Dokument enthaltenen Informationen von Zeit zu Zeit zu ändern. RIM ist jedoch nicht verpflichtet, die Benutzer über diese Änderungen, Updates, Verbesserungen oder Zusätze rechtzeitig bzw. überhaupt in Kenntnis zu setzen.

Diese Dokumentation enthält möglicherweise Verweise auf Informationsquellen, Hardware oder Software, Produkte oder Dienste, einschließlich Komponenten und Inhalte wie urheberrechtlich geschützte Inhalte und/oder Websites von Drittanbietern (nachfolgend "Drittprodukte und -dienste" genannt). RIM hat keinen Einfluss auf und übernimmt keine Haftung für Drittprodukte und -dienste, dies gilt u. a. für Inhalt, Genauigkeit, Einhaltung der Urheberrechtsgesetze, Kompatibilität, Leistung, Zuverlässigkeit, Rechtmäßigkeit, Schicklichkeit, Links oder andere Aspekte der Drittprodukte und -dienste. Der Einschluss eines Verweises auf Drittprodukte und -dienste in dieser Dokumentation impliziert in keiner Weise eine besondere Empfehlung der Drittprodukte und -dienste oder des Drittanbieters durch RIM.

SOFERN ES NICHT DURCH DAS IN IHREM RECHTSGEBIET GELTENDE RECHT AUSDRÜCKLICH UNTERSAGT IST, WERDEN HIERMIT SÄMTLICHE AUSDRÜCKLICHEN ODER KONKLUDENTEN BEDINGUNGEN, BILLIGUNGEN, GARANTIEN, ZUSICHERUNGEN ODER GEWÄHRLEISTUNGEN JEDER ART, EINSCHLIESSLICH, OHNE EINSCHRÄNKUNG, BEDINGUNGEN, BILLIGUNGEN, GARANTIEN, ZUSICHERUNGEN ODER GEWÄHRLEISTUNGEN HINSICHTLICH DER HALTBARKEIT, EIGNUNG FÜR EINEN BESTIMMTEN ZWECK ODER VERWENDUNGSZWECK, MARKTGÄNGIGKEIT, MARKTGÄNGIGEN QUALITÄT, NICHTVERLETZUNG VON RECHTEN DRITTER, ZUFRIEDENSTELLENDEN QUALITÄT ODER DES EIGENTUMSRECHTS ABGELEHNT. DIES GILT AUCH FÜR ZUSICHERUNGEN ODER GEWÄHRLEISTUNGEN, DIE SICH AUS EINEM GESETZ, EINER GEPFLOGENHEIT, USANCEN BZW. HANDELSGEPFLOGENHEITEN ERGEBEN ODER IM ZUSAMMENHANG MIT DER DOKUMENTATION ODER IHRER VERWENDUNG, DER LEISTUNG ODER MANGELNDEN LEISTUNG VON SOFTWARE, HARDWARE, DIENSTEN ODER DRITTANBIETER-PRODUKTEN UND -DIENSTEN STEHEN, AUF DIE HIER VERWIESEN WIRD. MÖGLICHERWEISE HABEN SIE ZUDEM ANDERE LANDESSPEZIFISCHE RECHTE. IN MANCHEN RECHTSGEBIETEN IST DER AUSSCHLUSS ODER DIE EINSCHRÄNKUNG KONKLUDENTER GEWÄHRLEISTUNGEN UND BEDINGUNGEN NICHT ZULÄSSIG. IN DEM GESETZLICH ZULÄSSIGEN UMFANG WERDEN SÄMTLICHE KONKLUDENTEN GEWÄHRLEISTUNGEN ODER BEDINGUNGEN IM ZUSAMMENHANG MIT DER DOKUMENTATION, DIE EINGESCHRÄNKT WERDEN KÖNNEN, SOFERN SIE NICHT WIE OBEN DARGELEGT AUSGESCHLOSSEN WERDEN KÖNNEN, HIERMIT AUF 90 TAGE AB DATUM DES ERWERBS DER DOKUMENTATION ODER DES ARTIKELS, AUF DEN SICH DIE FORDERUNG BEZIEHT, BESCHRÄNKT.

IN DEM DURCH DAS IN IHREM RECHTSGEBIET ANWENDBARE GESETZ MAXIMAL ZULÄSSIGEN AUSMASS HAFTET RIM UNTER KEINEN UMSTÄNDEN FÜR SCHÄDEN JEGLICHER ART, DIE IM ZUSAMMENHANG MIT DIESER DOKUMENTATION ODER IHRER VERWENDUNG, DER LEISTUNG ODER NICHTLEISTUNG JEGLICHER SOFTWARE, HARDWARE, DIENSTE ODER DRITTPRODUKTE UND -DIENSTE, AUF DIE HIER BEZUG GENOMMEN WIRD, STEHEN, EINSCHLIESSLICH, ABER NICHT BESCHRÄNKT AUF DIE FOLGENDEN SCHÄDEN: DIREKTE, VERSCHÄRFTEN SCHADENERSATZ NACH SICH ZIEHENDE, BEILÄUFIG ENTSTANDENE, INDIREKTE, KONKRETE, STRAFE EINSCHLIESSENDE SCHÄDEN, FOLGESCHÄDEN ODER SCHÄDEN, FÜR DIE ANSPRUCH AUF KOMPENSATORISCHEN SCHADENERSATZ BESTEHT, SCHÄDEN WEGEN ENTGANGENEN GEWINNEN ODER EINKOMMEN, NICHTREALISIERUNG ERWARTETER EINSPARUNGEN, BETRIEBSUNTERBRECHUNGEN, VERLUSTES GESCHÄFTLICHER DATEN, ENTGANGENER GESCHÄFTSCHANCEN ODER BESCHÄDIGUNG BZW. VERLUSTES VON DATEN, DES UNVERMÖGENS, DATEN ZU ÜBERTRAGEN ODER ZU EMPFANGEN, PROBLEMEN IM ZUSAMMENHANG MIT ANWENDUNGEN, DIE IN VERBINDUNG MIT RIM-PRODUKTEN UND -DIENSTEN VERWENDET WERDEN, KOSTEN VON AUSFALLZEITEN, NICHTVERWENDBARKEIT VON RIM-PRODUKTEN UND -DIENSTEN ODER TEILEN DAVON BZW. VON AIRTIME-DIENSTEN, KOSTEN VON ERSATZGÜTERN, DECKUNG, EINRICHTUNGEN ODER DIENSTEN, KAPITAL- ODER ANDERE VERMÖGENSSCHÄDEN, UNABHÄNGIG DAVON, OB SCHÄDEN DIESER ART ABZUSEHEN ODER NICHT ABZUSEHEN WAREN, UND AUCH DANN, WENN RIM AUF DIE MÖGLICHKEIT SOLCHER SCHÄDEN HINGEWIESEN WURDE.

IN DEM DURCH DAS IN IHREM RECHTSGEBIET ANWENDBARE GESETZ MAXIMAL ZULÄSSIGEN AUSMASS ÜBERNIMMT RIM KEINERLEI VERANTWORTUNG, VERPFLICHTUNG ODER HAFTUNG, SEI SIE VERTRAGLICHER, DELIKTRECHTLICHER ODER ANDERWEITIGER NATUR, EINSCHLIESSLICH DER HAFTUNG FÜR FAHRLÄSSIGKEIT UND DER DELIKTSHAFTUNG.

DIE IN DIESEM DOKUMENT GENANNTEN EINSCHRÄNKUNGEN, AUSSCHLÜSSE UND HAFTUNGSAUSSCHLÜSSE GELTEN (A) UNGEACHTET DER VON IHNEN ANGEFÜHRTEN KLAGEGRÜNDE, FORDERUNGEN ODER KLAGEN, EINSCHLIESSLICH, ABER NICHT BESCHRÄNKT AUF VERTRAGSBRUCH, FAHRLÄSSIGKEIT, ZIVILRECHTLICHER DELIKTE, DELIKTSHAFTUNG ODER SONSTIGE RECHTSTHEORIE UND SIND AUCH NACH EINEM WESENTLICHEN VERSTOSS BZW. EINEM FEHLENDEN GRUNDLEGENDEN ZWECK DIESER VEREINBARUNG ODER EINES DARIN ENTHALTENEN RECHTSBEHELFS WIRKSAM; UND GELTEN (B) FÜR RIM UND DIE ZUGEHÖRIGEN UNTERNEHMEN, RECHTSNACHFOLGER, BEVOLLMÄCHTIGTEN, VERTRETER, LIEFERANTEN (EINSCHLIESSLICH AIRTIME-DIENSTANBIETERN), AUTORISIERTE RIM-DISTRIBUTOREN (EBENFALLS EINSCHLIESSLICH AIRTIME-DIENSTANBIETERN) UND DIE JEWEILIGEN FÜHRUNGSKRÄFTE, ANGESTELLTEN UND UNABHÄNGIGEN AUFTRAGNEHMER.

ZUSÄTZLICH ZU DEN OBEN GENANNTEN EINSCHRÄNKUNGEN UND AUSSCHLÜSSEN HAFTEN DIE FÜHRUNGSKRÄFTE, ANGESTELLTEN, VERTRETER, DISTRIBUTOREN, LIEFERANTEN, UNABHÄNGIGEN AUFTRAGNEHMER VON RIM ODER RIM ANGEHÖRENDEN UNTERNEHMEN IN KEINER WEISE IM ZUSAMMENHANG MIT DER DOKUMENTATION.

Bevor Sie Drittprodukte bzw. -dienste abonnieren, installieren oder verwenden, müssen Sie sicherstellen, dass Ihr Airtime-Dienstanbieter sich mit der Unterstützung aller zugehörigen Funktionen einverstanden erklärt hat. Einige Airtime-Dienstanbieter bieten möglicherweise keine Internet-Browsing-Funktion in Zusammenhang mit einem Abonnement für den BlackBerry® Internet Service an. Erkundigen Sie sich bei Ihrem Service Provider bezüglich Verfügbarkeit, Roaming-Vereinbarungen, Service-Plänen und Funktionen. Für die Installation und Verwendung von Drittprodukten und -diensten mit den Produkten und Diensten von RIM sind möglicherweise Patent-, Marken-, Urheberrechts- oder andere Lizenzen erforderlich, damit die Rechte Dritter nicht verletzt werden. Es liegt in Ihrer Verantwortung, zu entscheiden, ob Sie Drittprodukte und -dienste verwenden möchten, und festzustellen, ob hierfür Lizenzen erforderlich sind. Für den Erwerb etwaiger Lizenzen sind Sie verantwortlich. Installieren oder verwenden Sie Drittprodukte und -dienste erst nach dem Erwerb aller erforderlichen Lizenzen. Alle Drittprodukte und -dienste, die Sie mit Produkten und Diensten von RIM erhalten, werden lediglich zu Ihrem Vorteil, ohne Mängelgewähr und ohne ausdrückliche oder konkludente Bedingungen, Billigungen, Garantien, Zusicherungen oder Gewährleistungen jeglicher Art von

RIM bereitgestellt, und RIM übernimmt in diesem Zusammenhang keine Haftung. Die Verwendung von Drittprodukten und -diensten unterliegt Ihrer Zustimmung zu den Bedingungen separater Lizenzen und anderer geltender Vereinbarungen mit Dritten, sofern sie nicht ausdrücklich von einer Lizenz oder anderen Vereinbarung mit RIM behandelt wird.

Bestimmte in diesem Dokument angeführte Funktionen erfordern eine Mindestversion von BlackBerry® Enterprise Server, BlackBerry® Desktop Software und/oder BlackBerry® Device Software.

Die Nutzungsbedingungen für RIM-Produkte und -Dienste werden in einer entsprechenden separaten Lizenz oder anderen Vereinbarung mit RIM dargelegt. KEINE DER IN DIESER DOKUMENTATION DARGELEGTEN BESTIMMUNGEN SETZEN IRGENDWELCHE AUSDRÜCKLICHEN SCHRIFTLICHEN VEREINBARUNGEN ODER GEWÄHRLEISTUNGEN VON RIM FÜR TEILE VON RIM-PRODUKTEN ODER -DIENSTEN AUSSER KRAFT.

Research In Motion Limited
295 Phillip Street
Waterloo, ON N2L 3W8
Canada

Research In Motion UK Limited
Centrum House
36 Station Road
Egham, Surrey TW20 9LF
United Kingdom

Veröffentlicht in Kanada